MASTER EN SCIENCES INFORMATIQUES

Un outil d'aide au diagnostic en homéopathie la méthode de Boenninghausen

Diagne, Papa Moussa

Award date:
1997

Année académique
1996-1997

Facultés Universitaires Notre-Dame de la Paix
Namur
Institut d'Informatique

Un outil d'aide au diagnostic en homéopathie: la méthode de BOENNINGHAUSEN

par DIAGNE Papa Moussa

Mémoire présenté
en vue de l'obtention du diplôme de « Maître en Informatique »

Promoteur : Professeur J. **FICHEFET**

Résumé

Dans ce mémoire, nous abordons le sujet d'aide au diagnostic en homéopathie, à savoir la méthode de Boenninghausen.

Le problème du choix correct du remède a toujours préoccupé Boenninghausen. Sa méthode est fondée sur la notion de symptôme complet. Un symptôme complet est un "grand symptôme" qui regroupe les quatre caractéristiques suivantes:

- ◊ Une localisation désignant le tissu, l'organe ou la fonction du corps ou du psychisme où se manifeste le symptôme.
- ◊ Une sensation désignant l'impression ou la conscience d'une impression par le système nerveux central, au départ dans cinq sens ou des voies sensitives.
- ◊ Une modalité concernant le moment, les circonstances et les conditions qui modifient un symptôme.
- ◊ Les symptômes concomitants survenant en même temps, et accompagnant la maladie.

Le symptôme complet comprend non seulement le ou les symptômes de la maladie mais les autres symptômes apparemment sans relation avec la maladie mais qui surviennent en même temps, c'est-à-dire les symptômes du malade.

Considéré à juste titre par Hahnemann comme le meilleur homéopathe de son époque, Boenninghausen a servi de référence à de nombreuses générations de médecins. Et pourtant, à l'époque actuelle, il semble presque complètement délaissé, certains ne le connaissent même pas de nom, suite à l'apparition du répertoire de Kent. Cet état de fait s'explique par la difficulté que représente le maniement des longues rubriques de son répertoire.

Mais aujourd'hui grâce à l'informatique, cette difficulté n'en sera plus une.

Nous avons conçu dans ce travail un outil de recherche, de répertorisation, de valorisation, et d'analyse, mais également un système de vérification qui assistera le médecin homéopathe dans sa tâche quotidienne.

Abstract

In this work, we discuss the subject of help to diagnosis in homeopathy, and more specifically Boenninghausen's method.

Boenninghausen has always felt concerned by the problem of the choice of the right remedy. His method is based on the notion of "complete symptom". A complete symptom is a "big symptom" that contains the four following characteristics ::

- ◊ A localisation stating the tissue, the organ or the function of the body or the mind, where the symptom is to be found..
- ◊ A sensation describing the impression or the consciousness of an impression in the central nervous system, captured by the five senses or other sensitive captors.
- ◊ A modality regarding the moment, the circumstances and the conditions that can modify a symptom.
- ◊ The concomitant symptoms that appear at the same time and accompany the disease.

The complete symptom includes not only the symptom(s) of the disease, but also the other symptoms appearing at the same time, apparently with no relation to the disease, in other words the symptoms of the patient.

Boenninghausen was considered by Hahnemann the best homeopath of his time and has been presented as a reference to numerous generations of doctors. However, nowadays, he has been almost forgotten, some do not even know his name; because of the, due to the publication of Kent's repertory and to the difficulty of manipulating the long rubrics of Boenninghausen's repertory.

But nowadays, thanks to computer science, this difficulty does not exist any more.

In this work, we have designed a tool for search, repertorization, valorisation and analysis, and a verification system as well, that will help the homeopath in his daily task.

Remerciements

Je remercie vivement toutes les personnes
qui m'ont permis de réaliser ce travail :

Monsieur J.Fichefet pour avoir accepter la direction de ce mémoire, et de son suivi régulier.

Dr G.Coquillart pour sa disponibilité, et ses précieuses explications sur Boenninghausen.

Monsieur P. Santantonio pour son assistance et ses conseils, notamment quant à l'utilisation du logiciel RADAR et ses ressources.

Monsieur P.Debuisson quant à l'apprentissage de XVT Design.

Tables des Matières

Introduction

1.1 OBJECTIFS

L'objectif de ce mémoire est d'élargir le système d'aide au diagnostic proposé dans RADAR et développé d'abord par l'asbl ARCHIMEDE puis par la sa ARCHIBEL. Ainsi RADAR devrait pouvoir offrir une autre approche dans le choix du remède, avec notamment la méthode de Boenninghausen. Cette nouvelle possibilité ne sera pas une entité isolée, mais s'inscrira plutôt dans le système informatique existant. Le tout, bien sûr, devra être un ensemble cohérent et harmonique.

Sur le plan informatique, nous devrons fournir un outil permettant une répertorisation, une valorisation, une vérification, et une analyse.

La répertorisation permettra au médecin d'enregistrer une liste de symptômes présentés par un patient.

La valorisation des symptômes enregistrés aboutira au tableau symptomatique du patient.

A cette étape, un module vérification sera envisager pour signaler l'état et la consistance du cas.

L'analyse devra suggérer au médecin un ou plusieurs remèdes.

Ce programme devra être un outil assez convivial, car étant voué à être utilisé par des médecins lors de leur consultation.

1.2 PLAN

Le mémoire est structuré comme suit:

Le contexte de ce travail étant l'homéopathie, nous tenons au chapitre 2, de présenter les différents concepts et principes généraux de cette discipline.

La méthode de Boenninghausen qui reste le moteur de notre application, sera vue en détails dans le chapitre 3.

Le chapitre 4 décrit et donne une idée assez intuitive de notre application dans sa réalisation.

Dans le chapitre 5, nous expliquons les possibilités existantes et offertes par RADAR. C'est l'occasion pour nous de définir et d'expliquer le choix des outils que nous allons utiliser pour cette application.

Le chapitre 6 est d'un aspect plus technique. En effet nous expliquons quelques algorithmes clefs de notre application. On trouvera aussi dans ce chapitre, l'architecture générale du programme, et ses points de communication avec RADAR.

Dans le chapitre 7 nous présenterons deux études cas.

Nous clôturons le travail par le chapitre 8 dans lequel nous formulons une conclusion générale.

Dans les annexes enfin, nous donnons une spécification complète et détaillée du programme.

Introduction à l'homéopathie

2. DEFINITION DE L'HOMEOPATHIE

2.1 ETYMOLOGIE

L'homéopathie est une méthode thérapeutique découverte il y a près de deux siècles. Bien que son objectif soit commun à celui de l'allopathie, l'homéopathie est une discipline très différente de la médecine traditionnelle. Les contrastes entre ces thérapies ne se bornent pas seulement aux méthodes et moyens de guérison, mais également sur le plan de la conception de l'organisme vivant.

Etymologiquement, le terme d'homéopathie vient du grec *omios*, " semblable ", et *pathos* " souffrant " et signifie " qui traite avec quelque chose produisant un effet semblable aux troubles dont il souffre. ".

Il a été signalé dans Platon et cité par Aristote dans ses " Grandes Morales ". C'est, dit Aristote :

> " L'état d'une âme qui ressent d'une façon semblable à la façon de sentir d'une autre âme et, qui par cela même, est plus disposée pour la véritable amitié.".

En d'autre terme c'est se mettre sous la peau d'un autre tout en lui voulant du bien.

2.2 DEFINITION MEDICALE DE L'HOMEOPATHIE

Nous dirons que l'homéopathie est la technique thérapeutique qui met en application la loi de la Similitude.

Si chacun s'accorde sur le fait que Hippocrate n'est pas l'inventeur de la médecine occidentale, nous devons cependant reconnaître qu'il en est en quelque sorte le fondateur en ce sens, qu'il est le premier écrivain médical qui nous soit parvenu. Dès cette époque, on admit qu'il existait souvent un parallélisme d'action entre le pouvoir toxicologique d'une substance et son action thérapeutique.

2.2.1 LA LOI DE LA SIMILITUDE ET L'EXPERIMENTATION

Toute substance capable de provoquer, chez un sujet sain et sensible, un ensemble de symptômes expérimentaux est susceptible de guérir un sujet malade qui présente un ensemble de symptômes semblables. Il ne s'agit pas d'un quelconque " principe" mais bien d'une loi de la nature au même titre que la loi de la gravitation universelle. L'observateur exempt de préjugés constate le phénomène mais ne l'explique pas. Une fois de plus, la nature a sa raison que la raison ignore. Cette formulation permet néanmoins au médecin d'utiliser cette loi pour soigner son malade comme l'ingénieur utilise la gravitation pour lancer des satellites. Il ne faut pas oublier que l'homéopathie a conquis ses titres de gloire en soignant le typhus pendant la guerre de Crimée, le choléra au cours de l'épidémie de Marseille et la diphtérie, toutes maladies qui ne sont pas particulièrement bégnines.

A la fin du XVIIIème siècle, vers 1760, S. Hahnemann, médecin saxon, avait déjà découvert, développé, systématisé les lois fondamentales de l'homéopathie. Il eut une renommée considérable parmi ses collègues pour ses travaux et publications remarquables dans les domaines de la médecine et de la chimie qu'il réalisait en marge de sa pratique médicale. Néanmoins, il restait insatisfait de la pratique de l'époque

> " Hahnemann était profondément perturbé par le manque de pensées fondamentales sous-tendant la thérapeutique de son époque qui consistait en des saignées, des purgations, des applications de sangsues, et l'utilisation de drogues toxiques."[VITH, p 95]

Pour cette raison, il s'est penché sur le problème et ne tarda pas à nous donner la voie qui allait raviver le feu de l'homéopathie. Il passera une bonne partie de sa vie à étudier avec le plus grand soin, sur l'individu sain, les symptômes propres à chaque substance, et ce, jusque dans les moindres nuances de leurs effets.

Il observa que le quinquina, par son action propre, produisait chez l'homme sain, une fièvre intermittente très analogue à celle que ce médicament guérit le mieux. Il en vint à penser que ces nuances seules peuvent servir, dans bien des cas, à caractériser l'action des médicaments, dont les symptômes violents se ressemblent plus ou moins presque tous. Il s'assura aussi que ce principe homéopathique se vérifiait également pour d'autres médicaments: que le mercure ne guérissait la syphilis que parce qu'il développait des symptômes analogues à ceux de cette maladie chez la personne saine qui en prenait; que le soufre ne guérissait la gale qu'à cause de la propriété dont il jouit de produire des éruptions cutanées analogues à celles de cette affection contagieuse, etc...

Ce ne fut d'abord qu'avec la plus grande circonspection que Hahnemann tenta sur ses malades l'application du principe qu'il avait redécouvert : il essaya de combattre les symptômes des maladies, en leur substituant en quelque sorte l'action de celle des substances déjà éprouvées qui offrait avec eux le plus d'analogie. Le succès couronna ses premières tentatives; il obtint des guérisons à la fois plus sûres, plus complètes et plus faciles.

Le second principe est la nécessité de l'expérimentation des agents thérapeutiques sur les individus sains afin d'en déterminer les vertus curatives.

L'administration à l'individu sain de substances médicamenteuses, provoque un ensemble de symptômes expérimentaux. Cet ensemble de symptômes représente, en fait, " une maladie artificielle" . La substance en expérimentation provoque une réaction globale de l'organisme qui dépend d'elle-même et de la capacité réactionnelle du sujet en expérience. Ce n'est que par la répétition des expérimentations que l'on arrive à se faire une idée des caractéristiques pharmacodynamiques de la substance. Signalons à ce sujet que l'expérimentation avec placebo[1] avait déjà été décrite dans l'Organon. Etonné par cette constatation, Hahnemann recommence l'expérience sur lui-même, sur les membres de sa famille, sur des confrères volontaires.

L'évidence de mille faits répétés, et dans lesquels le principe se vérifiait toujours, le conduisit à proclamer dans toute sa généralité, la loi de la similitude.

---

[1]Substance inactive substituée à un médicament pour étudier l'efficacité réelle de celui-ci en éliminant toute participation psychologique du malade

2.2.2 CONCEPT DE " DILUTION-DYNAMISATION"

L'expérience amena bientôt Hahnemann à une découverte nouvelle et très importante relativement au mode d'action des médicaments. Il commença tout d'abord, par réduire de beaucoup les doses usitées dans la médecine ordinaire. Ce qui fut, et est encore l'objet de tant de doutes. Le besoin d'une exactitude rigoureuse dans l'appréciation de quantités aussi exiguës, lui suggéra des procédés particuliers pour fractionner les doses:

> " Il soumet chacune de ses dilutions à une série de secousses vigoureuses (ou succussions, comme il les appelle), et découvre que les dilutions progressives obtenues de cette façon sont non seulement moins toxiques, mais encore plus puissantes.
>
> Il va considérer que l'eau distillée, l'alcool et le lactose (sucre de lait) sont des substances médicales inertes, et c'est avec elles qu'il dilue ses remèdes. Si le remède est soluble dans l'eau ou dans l'alcool, il mélange une part de substance à 99 parts de liquide, puis soumet le tout à 100 succussions vigoureuses. Cette solution " dynamisée " est appelée " première centésimale ". En mélangeant une part de cette première centésimale à 99 parts de liquide, agitée 100 fois à nouveau, il produit la deuxième centésimale la troisième étape de ce procédé, la substance originelle est donc diluée un million de fois. Puis elle le sera 100 millions de fois à la quatrième étape, et ainsi de suite. Apparemment Hahnemann pourra répéter cette opération 30 fois, mais ne dépassera pas ce nombre "[VITH, p 30]

Ce mode de préparation conduisit Hahnemann à cette singulière observation, que l'acte de broyer les substances, ou de secouer énergiquement les liquides qu'il mélangeait, développait à un haut degré, l'énergie de leur propriétés pharmaco-dynamiques.

2.3 LE PROBLEME PHYSICO-CHIMIQUE

Si l'expérience clinique montre de façon indiscutable que les substances ainsi traitées et prescrites suivant la Loi de la Similitude, provoquent, dans les mêmes conditions identiquement reproductibles, les effets curatifs désirés, tant sur l'homme, l'animal, que sur les végétaux, il faut aussi admettre que la technique de dilution-dynamisation amène à produire des solutions qui suivant la Loi d'Avogadro[2] ne contiendraient plus aucune molécule du principe supposé actif. La justification du pouvoir des remèdes ainsi obtenus dépasse les lois de la chimie et de la physique.

G. Vithoulkas précise en effet que:

> " Selon les lois de la chimie, il existe une limite au nombre de dilutions successives pouvant être réalisé sans perdre pour autant la substance originale. Cette limite est appelée " nombre D'Avogadro " et correspond approximativement à une dilution

[2]Avogadro est un chimiste italien né en 1776 et mort en 1856. Le nombre d'Avogadro est le nombre $N=6.023.10^{23}$ de molécules contenues dans une molécule-gramme.

> homéopathique de 24x (ou encore de 12c). En conséquence toute dilution excédant 24x ou 12c n'a pour ainsi dire pas la moindre chance de contenir fut-ce une molécule de la substance originale. " [VITH. P 103]

Toutefois, un tel raisonnement ne tient pas compte du procédé dynamique des succussions. Si la science ne trouve pas une justification rationnelle à un tel procédé, les explications théoriques ne manquent pas dans ce domaine.

Il semblerait que lors de l'action de chocs mécaniques visant à provoquer la succussion des molécules, il existerait une entité physico-chimique nouvelle, se traduisant par un nouvel agencement des molécules du solvant autour des ions.

Selon S. Hahnemann :

> " C'est dynamiquement, à la façon d'une contagion, que cette influence médicamenteuse se produit sur notre organisme, et cela, sans la moindre transmission de parcelle matérielle médicamenteuse." [VITH, p 32]

Selon G. Vithoulkas :

> " Il semble que cette technique libère une nouvelle forme d'énergie. L'énergie contenue, de façon illimitée, dans la substance originale est libérée et transmise aux molécules du solvant. Lorsqu'il n'y a plus de substance originale, l'énergie qui a été transmise au solvant peut néanmoins être augmentée *ad infinitum*. Ce sont les molécules du solvant qui ont alors capté l'énergie thérapeutique de la substance originale. Les résultats cliniques nous apprennent que l'énergie thérapeutique conserve toujours la " fréquence vibratoire " de la substance originale mais que cette énergie s'est accrue à tel point qu'elle est capable de stimuler le plan énergétique du patient pour entraîner la guérison. " [VITH, pp 104, 105]

Il ajoute toujours :

> " On a découvert qu'un électron peut subir l'influence d'un autre électron et qu'il se produit entre eux un échange informationnel, quelle que soit leur distance. Le physicien Richard Feynemann[3] considère que cela se produit par le truchement d'un échange de photons " virtuels " sans que " rien " ne se passe de façon concrète, sauf une interaction à distance. Il est maintenant démontré de façon expérimentale que deux particules, dans la mesure où elles ont été en contact à un moment donné, se garderont la mémoire de ce contact. Par le procédé dynamique des " succussions ", se crée-t-il les conditions électrostatiques optima permettant cet échange informationnel entre les électrons de la substance diluée et ceux de

[3]Selon l'encyclopédie Universalis, il s'agit du phisicien le plus remarquable du siècle; né New York le 11 mai 1918 et mort le 15 février 1988. En 1965, il reçut le prix Nobel de physique pour ses travaux sur l'electrodynamique quantique.

> la substance inerte qui lui sert de solvant? Il est permis de l'envisager (...) "[VITH2, pp 32, 33]

Pour Torwald Dethlefsen, tout repose sur le fait que c'est l'information qui guérit, et il précise d'ailleurs à ce propos:

> " L'information est toujours quelque chose d'immatériel qui a besoin d'un support pour être transmise. Ce support peut être fait de substances diverses (...) mais l'information qu'il porte reste la même, il remplit simplement un office. Les mêmes supports sont susceptibles de porter des informations très différentes, les mêmes informations peuvent être confiées à des supports très variés. En principe, la primauté est donnée à l'information et non au support. (...)
>
> Pour introduire un processus curatif, il faut (...) que la conscience se transforme, qu'elle augmente. Cela ne peut se produire que par l'apport d'une " information " nouvelle. C'est le rôle que prend le médicament censé apporter l'information manquante, il est alors vraiment un moyen de guérison, un " transmetteur d'information". "

Selon lui, les informations curatives se trouvent dans les minéraux, les plantes et les animaux dans lesquels elles s'individualisent. La dynamisation est alors un procédé qui débarrasse l'information de son attache organique et qui la lie à un " porteur d'information " (l'alcool, l'eau ou le lactose) afin de pouvoir la transmettre.

2.4 PATHOGENESIE - MATIERE MEDICALE

2.4.1 PATHOGENESIE

Si les médicaments semblent capables de guérir des symptômes analogues à ceux qu'ils peuvent produire, Hahnemann se devait de vérifier son hypothèse et de procéder à l'expérience de ceux-ci sur l'homme sain.

Tout d'abord, il convient de définir ce que l'on entend par individu sain. Bien sûr l'état de santé parfait n'existe pas; tout le monde a quelque chose, et si ce n'est pas physique, c'est moral. Il doit en être de même pour les chiens, chats, cobayes, lapins, souris, et autres animaux de laboratoire. Mais si la santé absolue est quelque chose de théorique, nous trouvons quand même quantité de gens qui ne se plaignent de rien, estiment se bien porter et qui, par-dessus le marché, sont satisfaits. Pour nous résumer, nous pourrions dire que la santé correspond à un état de bien-être physique, moral et social.

Dans le souci de clarifier et de systématiser le fruit de ses recherches, Hahnemann rédige pour chaque drogue, un protocole toxicologique très précis, des symptômes observés, auquel il donnera le nom de pathogénésie.

2.4.2 MATIERE MEDICALE

La Matière Médicale est constituée par l'ensemble des pathogénésies classées alphabétiquement. Hahnemann nous a laissé le premier traité de Matière Médicale pure.

2.5 METHODOLOGIE HOMEOPATHIQUE

2.5.1 NOTION DE SYMPTOMES

D'une façon générale les symptômes, c'est ce qui distingue un homme malade de son état de santé.

Pour le médecin homéopathe l'organisme vivant est d'abord un tout intégré; il n'est pas donc pas assimilé à un ensemble d'organes indépendants contrairement comme nous le laisse penser l'allopathie.

De plus, tout organisme vit en symbiose avec son environnement, il est affecté constamment et de diverses manières par l'environnement dynamique qui l'entoure. L'organisme s'ajuste ainsi continuellement de sorte à préserver un équilibre dynamique profitable à chacun. Tout déséquilibre correspond en effet à une destruction qui affaiblit aussi bien l'organisme que l'environnement dans lequel il vit.

Notons encore que dans la mesure où l'individu est envisagé comme un tout, un déséquilibre affecte l'être entier tant sur le plan physique que sur les plans émotionnel et mental.

Selon G. Vithoulkas chaque organisme est doté d'un mécanisme de défense naturelle contre les stimuli morbides:

> " Ce mécanisme est responsable du maintien d'un état d'homéostasie, c'est à dire d'un équilibre entre les processus tendant à désorganiser l'organisme et ceux tendant à préserver son organisation "

Dès lors, un rapport de force se crée entre le mécanisme de défense naturelle de l'organisme et la force des stimuli. Si cette dernière est plus importante, cela se traduit le plus souvent par des signes. G. Vithoulkas nous écrit à ce propos :

> " Lorsque la force des stimuli devient supérieure à la résistance de l'organisme, un déséquilibre se crée. (...) Bien que les effets de ce déséquilibre soient ressentis par la personne dans sa totalité, les manifestations, quant à elles s'expriment de façon privilégiée sur le plan physique, émotionnel ou mental selon la prédisposition individuelle. Ce sont ces symptômes ou groupe de symptômes qu'on a coutume d'appeler " maladie " alors qu'ils représentent en réalité la lutte du mécanisme de défense face à un stimuli morbide."

L'ensemble des symptômes détermine l'importance du déséquilibre, et le niveau auquel le plus de symptômes se sont manifestés constitue le centre de gravité de la maladie. Il y a amélioration de l'état de santé si le centre de gravité se déplace vers un autre de moindre importance, et régression de l'état de santé si le déplacement est inverse.

2.5.2 UNE PRESCRIPTION HOMEOPATHIQUE

Connaissant en détail la matière médicale homéopathique, le médecin compétent dans cette branche de l'Art, va considérer l'ensemble des symptômes présentés par le malade. La collecte de symptômes est le problème le plus difficile de la médecine en général. La

pratique de l'homéopathie nous apprend que le malade s'exprime uniquement par des symptômes.

Les remèdes lorsqu'ils sont également testés sur des individus sains, produisent aussi un ensemble de symptômes particuliers à chaque substance médicamenteuse. Et lorsque le tableau symptomatique du malade et celui du remède coïncident, si celui-ci est administré à la dose requise, les symptômes de la maladie disparaissent et le malade retrouve son état de santé.

Dans la clinique journalière, l'homéopathe devra procéder à une individualisation à deux niveaux différents:

- celui du malade: collecte de la totalité des symptômes
- celui du remède : trouver dans la nature la drogue capable de provoquer la maladie de substitution la plus semblable à celle présentée par le malade.

La méthode de Boenninghausen

3. LA METHODE DE BOENNINGHAUSEN

3.1 INTRODUCTION

En 1976, Robert Perry, directeur des Laboratoires Homéopathiques de France, décidait, à l'occasion du cinquantenaire du Laboratoire, de publier en fac-similé le célèbre répertoire de Boenninghausen. Le titre exact est : " Manuel de thérapeutique homéopathique pour servir de guide au lit des malades et à l'étude de la matière médicale pure ". Il écrivait dans l'Avant-propos :

> " Que le futur de toute vérité authentique trouve profit à se nourrir de son passé. "

Né en 1785 et mort en 1864, Boenninghausen est un contemporain de Hahnemann, Jahr, Hering. Il fait partie des très grands de l'âge d'or de l'homéopathie. Esprit brillant, il est nommé à 20 ans, conseiller d'Etat par Louis Bonaparte, Roi de Hollande. Par la suite il fera une carrière dans la Haute Administration de Westphalie et s'occupera d'agriculture, de remembrement, d'élevage, et de botanique. Sa compétence est telle qu'une plante portera son nom la " Boenninghausenie".

En 1828, il est atteint d'une phtysie purulente et guéri par l'homéopathie du docteur Wheil. Il comprend alors tout l'intérêt de cette nouvelle médecine et décide de lui consacrer le reste de son existence qu'il lui doit. Il apprend la médecine et l'homéopathie, entre en contact et correspond avec les maîtres de l'époque : Hahnemann, bien sûr, qui le tiendra en si haute estime qu'il n'hésitera pas à lui demander son avis sur de nombreux sujets, mais aussi, Jahr et Hering, pour ne citer que ceux-ci. Il écrira de nombreux articles dans différentes revues et sera nommé Docteur en Médecine sans passer d'examen.

En 1832 il publie le " répertoire des remèdes anti-psoriques " et en 1835 " le répertoire des remèdes qui ne sont pas anti-psoriques ". Ces ouvrages, résultat de la compilation d'un néophyte ne le satisferont pas et avec les encouragements de Hahnemann, il met en chantier son Manuel qui sera publié en 1845 à Munster et traduit aussitôt en français par le docteur Roth et en anglais par Allen, l'auteur de l'Encyclopédie de Matière Médicale : le " Boenninghausen's Therapeutic Pocket Book " qui servira de base à ce mémoire.

3.2 POURQUOI BOENNINGHAUSEN ?

Il y a dix ans déjà le docteur G. Coquillart[4] écrivait dans son article "La similitude selon Hhnemann" :

> " Dix-sept ans de pratique uniciste grâce à l'utilisation du répertoire de Kent[5] ne m'ont pas empêché de rechercher d'autres

[4]Dr. Guy Coquillart médecin homéopathe membre de la sociétére belge d'homéopathie (IBET) remarqué par ses recherches et publications sur Boenninghausen.

> voies dans la recherche du *similimum*. (...) Boenninghausen je l'ai " rencontré " lors d'un séminaire de l'I.B.E.T. Il a établi un système original qui permet de trouver rapidement et avec un maximum de sûreté le remède correct. "

Considéré à juste titre par Hahnemann comme le meilleur homéopathe de son époque, Boenninghausen a servi de référence à de nombreuses générations de médecins. Et pourtant, à l'époque actuelle, il semble presque complètement délaissé, certains ne le connaissent même pas de nom, suite à l'apparition du répertoire de Kent. Cet état de fait s'explique par la difficulté que représente le maniement des longues rubriques de son répertoire.

Le problème du choix correct du remède a toujours préoccupé Boenninghausen. Il s'en explique :

> " Les défauts des répertoires publiés jusqu'à ce jour proviennent, selon moi, principalement de ce que les auteurs se sont renfermés dans les limites de la matière médicale pure, en y rattachant sans doute les faits bien constatés dans la pratique, mais sans aller jusqu'à les mettre à profit toutes deux pour apprécier la valeur de chaque symptôme, pour les compléter l'une par l'autre, et pour aider à combler les lacunes qui frappent à chaque instant le praticien. "

3.3 ORIGINALITE DE LA METHODE DE BOENNINGHAUSEN

Tout d'abord il est bon de rappeler que l'ensemble des symptômes traduit la réaction globale du sujet à une cause connue ou non connue. Cette réaction est globale, elle est fonction de la cause tout autant que des potentialités du sujet. La maladie n'est qu'un groupement caractéristique de symptômes qui permet au médecin d'affirmer que ceci est une rougeole, ceci est une appendicite... Ce terme de " maladie ", qu'Hahnemann brocarde pourtant volontiers, est essentiel pour comprendre la suite du raisonnement. D'ailleurs tout au long de l'Organon, Hahnemann parle de maladie et il définit la thérapeutique homéopathique par le fait qu'elle oppose une maladie artificielle à une maladie naturelle. Dans le paragraphe 153, il nous apprend que :

> " La comparaison de l'ensemble des symptômes de la maladie naturelle avec la liste des symptômes pathogénésiques de médicaments bien expérimentés, est, il est utile de le répéter, la condition *sine qua non* pour trouver, parmi ces derniers, une puissance pharmaco-dynamique similaire au mal à guérir."

La maladie naturelle est définie par le groupement caractéristique des symptômes, il reste au médecin à définir le groupement caractéristique des symptômes de la maladie artificielle. Ceci ne sera possible que par une connaissance approfondie de la Matière Médicale Pure, par l'aide, et l'expérience qu'ont laissé certains homéopathes. C'est la justification des innombrables livres de matière médicale qui mettent en évidence tels ou tels aspects des remèdes et qui ont tous leur utilité.

---

[5]James Tyler Kent médecin homéopathe du XIX^{ème} siècle. La première édition de son répertoire date de 1897.

Ce travail permet au médecin de percevoir ce qui est ou non important dans le fatras symptomatique du malade d'une part et de la matière médicale d'autre part.

La maladie est donc vue comme un groupement caractéristique de symptômes. Et des symptômes banals deviennent caractéristiques par leur regroupement. C'est ici qu'apparaît tout le génie de Boenninghausen. A propos des symptômes caractéristiques, Hahnemann ajoute en note :

> " Le conseiller d'Etat, Baron Von Boenninghausen a rendu un grand service à l'homéopathie par la publication de son répertoire contenant les symptômes caractéristiques des remèdes homéopathiques, ainsi que G.H.G. Jahr dans son manuel des symptômes principaux, édité maintenant pour la troisième fois sous le titre de " *Grand Manuel* ". "

Au paragraphe 154, il enfonce encore le clou:

> " Plus la contre-image, composée avec le groupe des symptômes pathogénésiques du remède qui paraît mériter la préférence, renfermera des symptômes semblables à ceux, caractéristiques, frappants, originaux, inusités et personnels de la maladie naturelle, plus la ressemblance réciproque sera parfaite et plus aussi ce remède sera convenable, homéopathique: le spécifique en la circonstance."

3.3.1 LE SYMPTOME COMPLET

Un symptôme représente plus qu'un simple fait. En effet c'est un fait qui a une histoire, une localisation, et qui évolue selon une direction et des conditions particulières.

Boenninghausen nous dit, dans la préface de son Manuel, page XIV, en parlant des lacunes de la matière médicale pure:

> " Si un grand nombre de symptômes sont incomplets, c'est parce qu'on n'indique pas avec précision la partie du corps où ils se sont manifestés, ni la nature des sensations, et surtout parce qu'on ne signale pas le moment, le lieu et les circonstances de l'aggravation ou de l'amélioration, les difficultés que rencontre leur juste appréciation dans le traitement d'une maladie redoublent encore par l'impossibilité d'en découvrir le caractère principal dans un seul symptôme, quelque complet qu'il soit."

- La localisation désigne le tissu, l'organe ou la fonction du corps ou du psychisme où se manifeste le symptôme.
- La sensation désigne l'impression ou la conscience d'une impression par le système nerveux central, au départ des cinq sens ou des voies sensitives. On distingue toutefois les sensations qui ont une base avec les symptômes (comme une qualifications de la douleur) et les sensations sans fondement anatomo-clinique (les illusions mentales ou physiques).
- Les modalités concernent le moment, les circonstances et les conditions qui modifient un symptôme. Parmi elles, les modalités d'aggravations et d'amélioration révèlent une importance particulière.

Ces trois éléments sont plus connus sous le nom de "Trépied de Hering ".

Un symptôme, quelque complet qu'il soit, ne permet pas encore à lui seul, de caractériser un remède ni un cas donné de maladie. Pour déterminer le caractère spécifique d'un remède dans un cas donné de maladie, il faut donc plusieurs symptômes. L'ennui est que, tant dans la matière médicale pure que dans les recueils d'observations des malades il est rarissime de trouver un symptôme complet.

Le docteur Jean Claude Grégoire nous dit à ce propos :

> " Le portrait symptomatique du remède tel qu'il ressort du proving[6], est semblable à un portrait d'ancêtre qui aurait été mangé par des mites: il est plein de trou. Si l'on veut pouvoir comparer l'ancêtre avec son portrait, il faut donc par la pensée reconstituer, ce qui manque au tableau, ce qui a disparu dans les trous.
>
> L'opération de l'esprit qui exécute ce genre de prouesse est nécessairement une généralisation, aux endroits qui font défaut (...) "

3.3.2 LA GENERALISATION

Le génie de Boenninghausen est, en fait son aptitude à la " généralisation " et l'originalité de son travail réside dans le fait que, contrairement aux répertoires analytiques habituels, il s'agit ici d'un répertoire des caractéristiques des remèdes.

Boenninghausen nous révèle, toujours dans la préface page XX:

> "Mon ami le docteur Lutterbeck donna à mon absence à un de mes malades (que je lui confie toujours en pareil cas) contre quelques restes d'une phtisie tuberculeuse dont je l'avais guéri, notamment contre un poli désagréable des dents avec mucosité abondante, s'exacerbant considérablement pendant deux jours, toutes les fois qu'il se faisait la barbe, avec le plus grand succès, quoique le seul symptôme de la peau de la face, observé par le docteur Adams, n'existât pas, et surtout que ce symptôme d'exacerbation n'eût pas été une seule fois parfaitement constaté."

Nous l'aurons compris, le docteur Lutterbeck a donc dans ce cas précis d'exacerbation après rasage, prescrit le remède sur la modalité d'aggravation bien que dans le proving, elle s'appliquait à un symptôme différent de celui que présentait le malade.

Par analogie une modalité observée dans le proving pour une localisation précise a été étendue à une région voisine.

Et Roberts[7] parlant de Boenninghausen dans son " Introduction to The Therapeutic Pocket Book ", nous apprend à la page 9 :

> " Il apprit bientôt que l'on pouvait, par analogie, compléter d'une manière fiable tel symptôme qui existait à l'état incomplet dans une région quelconque d'un cas, en observant les modalités des autres régions de ce cas. Si par exemple, il n'était pas possible, en

[6]La méthode par laquelle les drogues sont expérimentées sur des organismes sains

[7]Roberts dans son ouvrage qu'il a publié en 1935, à Philadelphia, chez Boericke & Tafel: "The principles and practicability of Boenninghausen's therapeutic pocket book"

> questionnant un patient, de préciser ce qui aggravait ou améliorait un symptôme particulier de son cas, il était habituel que le malade exprime facilement une modalité d'amélioration de quelque autre symptôme. Il ne lui fallut pas longtemps pour découvrir que les modalités d'aggravation ou d'amélioration n'étaient pas limitées à tels ou tels symptômes particuliers, mais qu'au contraire, comme le fil rouge dans les cordages de la Royal Navy, elles s'appliquaient à tous les symptômes du cas. "

Boenninghausen s'appuie sur la théorie hahnemannienne que c'est le patient qui est malade, et non pas sa tête, ni son oeil, ni son coeur. Chaque symptôme qui se rapporte à une partie du corps peut être attribué à l'homme tout entier.

Un symptôme complet donne une bonne idée de la manière de réagir d'un individu malade, et le répertoire des caractéristiques de Boenninghausen permet déjà rapidement, grâce à l'Informatique, de trouver un certain nombre de remèdes possédant ce symptôme dans leur pathogénésie.

Le choix s'affinera par la recherche d'autres symptômes complets et surtout par la découverte du " symptôme concomitant ".

3.3.3 SYMPTOMES CONCOMITANTS

Concomitant signifie " Qui est, qui survient avec; c'est-à-dire circonstance qui accompagne ". Les symptômes concomitants surviennent en même temps, accompagnent la maladie. Ils peuvent ne pas avoir de rapport, au sens physiopathologique, avec les symptômes de la maladie; ils ont une relation de temps et circonstance avec eux.

C'est la base de la technique de Boenninghausen. Les symptômes concomitants permettent la " personnalisation ", l'individualisation du cas. Il s'agit, en fait, d'une reconnaissance de personnage ; le personnage fictif qu'est le médicament avec le personnage réel qu'est le malade. Et nous savons tous par expérience que même un myope reconnaît ou croit reconnaître de loin un personnage familier. Comment est-ce possible, sinon par la perception des grandes caractéristiques de ce personnage. Ce n'est qu'en se rapprochant qu'il distinguera d'éventuelles différences qui viendront infirmer cette reconnaissance. Il n'en sera pas moins vrai qu'il existe, dans ce cas, une ressemblance plus ou moins marquée. Il s'agit d'une ressemblance qui ne deviendra *similimum* que par l'absence de plus ressemblant qu'elle.

Pour Boenninghausen, comme pour Hahnemann qui l'écrit dans le paragraphe 210 de l'Organon :

> " Dans la thérapeutique et dans n'importe quel cas de maladies, le moral du malade est à relever comme un élément parmi les plus importants dans la totalité des symptômes."

Et il ajoute au paragraphe 211 :

> " Cela va si loin que l'état moral du malade devient souvent, dans la sélection du remède homéopathique, l'élément le plus déterminant, parce qu'il constitue une des manifestations les plus caractéristiques et les plus essentielles de celles qui, entre toutes,

> doivent le moins échapper au médecin habitué à faire des observations exactes. "

Ce point est très important car il montre que la maladie " modifie " le comportement du malade. Il ne s'agit pas d'un état habituel.

Les symptômes concomitants permettent de choisir parmi le groupe de remèdes préalablement sélectionnés celui qui est le plus individualisé. Voyons ce que Boenninghausen nous dit dans la préface du Manuel pp XVIII à XIX:

> " Je dois donner quelques explications sur la rubrique des symptômes concomitants. Convaincu de l'importance des symptômes qui se manifestent en même temps que d'autres, et forment avec eux un groupe, j'ai recueilli depuis plusieurs années tout ce qui appartient aux symptômes accessoires signalés dans la matière médicale pure, toutes les données de ma propre expérience et de l'expérience des autres, et le nombre s'en est tellement accru que j'ai été en état d'en tirer des règles générales. Ces règles prouvent avec une grande évidence qu'un médicament répond mieux qu'un autre aux symptômes accessoires qui se manifestent avant, pendant ou après le mal primitif, mais en même temps que ces symptômes accessoires ne consistent pas exclusivement en tels ou tels accidents; qu'au contraire, toute espèce de douleurs rentrant dans la sphère d'action du médicament peut se joindre à la maladie, bien que les symptômes caractéristiques s'offrent plus fréquemment que les autres. "

Hahnneman partage cette idée de Boenninghausen, il écrit dans l'Organon au paragraphe 153 :

> " Il faut surtout et presqu'exclusivement, dans la recherche du remède homéopathique, s'attacher aux symptômes objectifs et subjectifs caractéristiques les plus frappants, les plus originaux, les plus inusités et les plus personnels. Ce sont ceux-là principalement, qui doivent correspondre aux symptômes très semblables du groupe appartenant au remède à trouver, pour que ce dernier soit celui qui convienne le mieux à la guérison."

Les symptômes concomitants peuvent donc être, soit des symptômes coexistants, soit des symptômes ayant une relation de temps ou de circonstance, soit des symptômes d'aggravation ou d'amélioration, soit des modifications d'humeur.

Ceci indique clairement que les symptômes qui serviront au diagnostic ne sont pas seuls à devoir être pris en considération. C'est donc bien au dépend des symptômes individuels que se construit la totalité des symptômes.

3.3.4 TOTALITE DES SYMPTOMES

Nous avons vu que si les symptômes groupés selon un ensemble cohérent pouvaient définir une maladie, cet ensemble est insuffisant pour englober à la fois la maladie et la manière dont le malade fait sa maladie. Nous ne traitons pas une entité nommée rougeole mais la rougeole de Monsieur X.

La totalité est l'image complète de la maladie. C'est un grand symptôme qui comprend non seulement le ou les symptômes complets de la maladie mais les autres symptômes apparemment sans relation avec la maladie mais qui surviennent en même temps, c'est à dire les symptômes du malade.

En résumé comme l'a si bien dit le docteur G. Coquillart:

> " Les symptômes concomitants sont à la totalité ce que les modalités sont au symptôme complet. La totalité des symptômes est à la maladie ce que la personne est à l'organisme."

3.4 ANALYSE DE LA METHODE

Boenninghausen fut d'abord un juriste avant de devenir botaniste, puis un des plus célèbres homéopathes. Influencé par sa première profession, il a appliqué à la découverte du remède *similimum* les règles déjà étudiées par Quintilien, avocat romain du premier siècle de notre ère. Comme lors d'une instruction criminelle Boenninghausen se pose les sept questions fondamentales.

3.4.1 QUI ?

Qui donc est ce malade ? Boenninghausen nous dit:

> " Il est évident qu'il faut placer au tout premier plan du tableau de la maladie la personnalité, l'individualité du malade, car c'est sur elle que repose le comportement naturel du patient.
>
> Cette personnalité est constituée en premier lieu du sexe et de l'âge; ensuite de la constitution corporelle et du tempérament ; dans les deux cas il faut, si possible, distinguer les différences, pour autant qu'il en apparaisse d'appréciables, entre l'état de maladie et celui de bonne santé. Pour toutes ces particularités, tout ce qui diffère à peine ou pas du tout de l'état naturel habituel du malade n'offre que peu d'intérêt, mais tout ce qui s'en écarte d'une manière frappante ou inhabituelle en méritera d'autant plus notre attention. On trouvera ici les modifications les plus grandes et les plus importantes, principalement dans les états de l'âme et de l'esprit, qu'il faudra scruter avec la plus grande attention, non seulement si elles sont marquées, mais encore si elles se produisent rarement et que, pour cela, elles ne correspondent qu'à peu de remèdes. Dans tous les cas de ce genre, il y a d'autant plus de raisons d'analyser ces états avec toute l'exactitude possible, que l'affectation corporelle recule alors à l'arrière-plan, et que, précisément pour ce motif, elle nous offre peu de points à saisir pour être à même de faire un choix sûr entre les remèdes concurrents.
>
> Le paragraphe 104 de l'Organon fait un devoir à l'homéopathe d'établir une description écrite du tableau de la maladie. Une fois qu'il y aura acquis une certaine aisance, le médecin saura répondre sans difficulté à cette exigence (de saisir la personnalité du malade) et qui s'avérera d'une utilité sans cesse croissante. Car, puisque

> chaque être humain présente une nature individuelle différente de celle de toute autre personne, et puisque chaque remède doit être exactement adapté à cette individualité, en concordance avec les symptômes qu'il est susceptible de produire dans l'homme total, il s'impose immédiatement que la toute première investigation doit se rapporter au " *Quis* ". On rejettera un très grand nombre de remèdes, simplement parce qu'ils ne correspondent pas à la personnalité.
>
> L'individualité de l'esprit et du comportement du patient nous fournissent ici les points les plus importants, et le plus souvent presque les seuls points décisifs pour le choix du remède, lorsque la maladie en question est une maladie de l'âme ou de l'esprit (...)."

Si nous avons cité un si long paragraphe c'est pour bien montrer et faire ressortir l'importance prépondérante que Boenninghausen accorde aux symptômes mentaux.

Différencier les particularités qui entrent dans l'état naturel habituel du patient et toutes celles qui s'en écartent, n'est pas chose facile. C'est un point qui demande de la patience et le savoir-faire du praticien.

3.4.2 Quoi ?

Cette question se rapporte évidemment à la maladie, c'est à dire à sa nature et à ses particularités.

Boenninghausen insiste ici sur le fait qu'il est impossible de se baser, pour la prescription, sur les diagnostics nosologiques de l'allopathie, et que finalement

> " Un diagnostic pourra tout au plus, et encore, pas toujours, servir à exclure de la compétition les remèdes qui ne correspondent pas au génie de la maladie, mais qui semblent agir principalement sur d'autres parties de l'organisme. "

3.4.3 Ou ?

En fait le siège de la maladie fait partie du point précédent, mais néanmoins il mérite qu'on le mette plus particulièrement en valeur, parce qu'il fournit fréquemment un symptôme caractéristique, étant donné que chaque remède agit davantage, et également d'une manière plus marquée, sur certaines régions particulières de l'organisme vivant.

Boenninghausen n'exclut pas, dans cette partie, l'utilisation de moyens et techniques dont dispose la médecine moderne pour établir un diagnostic précis:

> " Si Hahneman et ses élèves avaient aussi bien connu que nos jeunes confrères la pratique de l'auscultation et de la percussion, ainsi que l'usage du stéthoscope, du plessimètre,... il ne fait aucun doute qu'ils en auraient fait l'usage le plus étendu, pour arriver à une connaissance et à une délimitation plus exactes des affectations internes. Ils auraient découvert, par exemple dans les maladies pulmonaires, des signes locaux précis indiquant l'usage de certains remèdes, et ils auraient posé les indications avec plus de précision,

en ne se limitant pas à dire qu'il s'agit du côté gauche ou droit, ou du sommet ou de la base (...). "

Mais Boenninghausen nous met en garde quant à l'utilisation de ces instruments, ou plutôt de l'interprétation qui s'en suivrait:

" Enfin il nous faut savoir à ce propos que ni les changements internes, que l'on peut percevoir au moyen de ces instruments, ni les changements matériels externes, qui se manifestent ouvertement à notre attention, ne représentent jamais la maladie dynamique elle-même, mais qu'ils n'en sont que les produits, et qu'ils ne se développent que dans le cours de la maladie. C'est pour cette raison que, lorsqu'on maîtrise au moyen du remède approprié, les premiers commencements de la maladie avant que ces désorganisations ne prennent place, ces dernières n'arrivent pas à se développer. Ce serait un procédé inexcusable de permettre à ces maladies de progresser jusqu'au point où l'on peut reconnaître des changements matériels au moyen d'instruments. Il était nécessaire que ceci fût dit, en passant, afin de montrer comment procède l'homéopathe (...)."

3.4.4 PAR QUELS MOYENS ? AVEC QUELS COMPLICES ?

Il est manifeste qu'il s'agit des symptômes d'accompagnement. Boenninghausen nous en dit :

" Et bien puisqu'en homéopathie le but principal consiste à établir quel est le remède qui correspond le plus complètement à la totalité des symptômes, il est évident que ce point est de la plus grande importance et mérite d'être considéré de la manière la plus soigneuse.

En effet, chaque maladie présente, dans ses phénomènes reconnaissables, un nombre plus ou moins grand de symptômes, et c'est seulement leur totalité qui représente son tableau complet. On peut comparer ce tableau à un portrait, qui ne pourra prétendre à une ressemblance frappante que s'il présente fidèlement tous les traits de l'original. Il ne suffit pas que l'on ait représenté la bouche, le nez, les oreilles, etc. d'une manière telle qu'elle ne soit caractéristique que de l'espèce humaine, le distinguant ainsi du singe et d'autres animaux mais, puisque chaque physionomie possède ses propres particularités qui la distinguent de toute autre, il faudra donc représenter également ici les anomalies plus ou moins prononcées, avec le plus de vérité et de fidélité possible, il faudra leur donner leur importance respective. Les parties secondaires elles aussi, qui en quelque sorte forment l'arrière-plan, doivent représenter un tout, comme il existe en réalité, de façon à fournir une ressemblance parfaite.

C'est de ce point de vue qu'il faut voir les souffrances concomitantes lorsque nous choisissons un remède conformément à la devise " SIMILIA SIMILIBUS ". Il est évident, pour cette raison,

> que les symptômes rares, frappants et caractéristiques qui se présentent réclament d'être mieux mis en valeur que les symptômes communs, parce que c'est d'eux principalement, mais pas exclusivement que dépend la similitude."

Il découle de ceci, naturellement, que la valeur de ces symptômes concomitants par rapport au but poursuivi, varie dans une large mesure.

En tout premier lieu, les symptômes que l'on retrouve dans presque toutes les maladies peuvent être négligés, à moins qu'ils ne se manifestent d'une manière frappante.

La même chose vaut pour toutes les souffrances qui ont coutume de se manifester comme des concomitants constants, ou du moins habituels, dans la maladie en question, à moins qu'ils ne se distinguent par quelque particularité rare et qu'ils n'apportent ainsi quelque élément caractéristique.

Par ailleurs, il faut soigneusement noter tous les symptômes d'accompagnement

- ◊ Ceux qui se montrent rarement en corrélation avec la maladie principale et que, pour la même raison, l'on trouvera rarement également dans les expérimentations.
- ◊ Ceux qui appartiennent à une autre sphère de la maladie que la plainte principale.
- ◊ Et enfin ceux qui possèdent plus ou moins les signes caractéristiques d'un remède, même si, jusqu'à présent, on n'a pas encore remarqué une telle juxtaposition de symptômes.

Selon Boenninghausen, les symptômes concomitants cités ci-dessus méritent une attention particulière, car il pourrait s'agir de symptômes caractéristiques. A ce propos Il nous apprend :

> " Si en outre, parmi les symptômes concomitants, il devait y en avoir l'un ou l'autre qui dessine le portrait d'un remède d'une façon claire et si précise que son indication en deviendrait évidente, cet unique symptôme acquerrait par là une importance telle qu'il surpasserait même les symptômes de la plainte principale, et que l'on pourrait dès lors regarder immédiatement ce remède comme le plus convenable(...).
>
> Il y a ici un point qui mérite une mention spéciale, car il montre particulièrement l'importance et la valeur des symptômes concomitants: sachez que l'on a découvert plusieurs remèdes très efficaces, et en partie spécifiques, pour certaines maladies, presqu'exclusivement par le biais des symptômes concomitants, car les autres symptômes, qui indiquaient la maladie principale, n'avaient donné aucun indice en ce sens, et d'ailleurs ils n'auraient absolument pas pu le faire, puisqu'il est impossible que les signes les plus immédiatement évidents de la maladie puissent jamais donner une indication sur le caractère réellement individuel de cette affection. (...) "

Boenninghausen oppose l'homéopathie à l'allopathie qui donne un remède seulement contre la plainte principale.

3.4.5 POURQUOI ?

Boenninghausen distingue:

- Les causes internes, qui se rapportent à la disposition générale naturelle.
- Les causes externes ou occasionnelles, qu'il répartit en:
  - ◊ Causes miasmatiques, psore, syphilis et sycose.
  - ◊ Empoisonnements et maladies médicamenteuses.
  - ◊ Traumatisme.
  - ◊ Acteurs climatiques (chaleur, froid, etc.).
  - ◊ Contagion, dans le cas des maladies infectieuses. Dans ce cas Boenninghausen souligne que :

> " Lorsque dans une famille, nous trouvons un cas de fièvre typhoïde infectieuse, c'est précisément le remède que l'on a administré au malade en accord avec ses symptômes qui protégera de la manière la plus sûre le reste de la maisonnée contre l'infection, car il détruit la disposition naturelle pour cette maladie (...)".

3.4.6 COMMENT ?

De par son étymologie, cet adverbe décrit d'une manière excellente l'essence et la portée de cette question. Le mot *Modus*, en effet, chez les classiques anciens, ne se rapporte pas seulement à la manière et au mode en général, mais encore à toutes les modifications qui peuvent se produire dans quelque chose. Tout ce qui à l'exception du moment, qui fait partie de notre dernière question (*Quando*), possède le pouvoir de produire une modification, aggravation ou amélioration, chez le patient, appartient naturellement à cette rubrique.

Boenninghausen place ici les désirs et aversions alimentaires. Il souligne aussi le fait qu'il faudra mettre en doute l'applicabilité d'un remède qui aurait paru indiqué par les cinq points précédents, mais qui auraient des modalités contraires à celles du malade.

Il termine ce point en affirmant qu'il " considère que les indications fournies par cette question et la suivante sont les plus importantes dont on peut le moins douter, et par conséquent les plus décisives" pour le choix du remède.

3.4.7 QUAND ?

Cette dernière question concerne l'apparition, de l'aggravation ou de l'amélioration des souffrances, et elle ne le cède en rien à la précédente.

Ici encore deux points ont un effet immédiat sur le choix du remède, à savoir:

- Le retour périodique des symptômes morbides après une interruption plus ou moins longue.
- Les aggravations et améliorations qui dépendent de l'heure de la journée.

> " Le retour périodique des phénomènes morbides coïncide souvent avec des (...) causes occasionnelles particulières, parmi

lesquelles il faut relever les souffrances menstruelles, ainsi que celles qui sont rythmées par les saisons, le temps qu'il fait, etc. Lorsqu'il n'est pas possible de découvrir de telles causes secondaires et lorsque, ce qui est le cas le plus fréquent, les accès ne sont pas étroitement liés à des périodes définies avec précision, ces causes n'ont aucune valeur pour les homéopathes (...).

Mais les aggravations et améliorations à des heures particulières de la journée sont d'une plus grande importance, et ceci autant pour celles qui se rapportent à des symptômes isolés que pour celles qui se rapportent à la santé en général. "

3.5 Le "Manuel" et le Repertoire de Boennighausen

3.5.1 Le "Manuel"

C'est avec les encouragements d'Hahnemann, qui connaissait ses travaux, que Boenninghausen entreprit la rédaction de son manuel. Dans sa préface Boenninghausen affirme l'impossibilité de:

" Découvrir le caractère principal (du médicament) dans un seul symptôme, quelque complet qu'il soit."

Et surtout :

" La nécessité absolue de recourir à la matière médicale pure pour juger de la valeur de la plupart des symptômes par la comparaison attentive de l'ensemble des phénomènes."

Il explique plus loin:

" Je m'aperçus que, très vraisemblablement j'atteindrais le but d'une manière plus simple et plus complète à la fois, si, en faisant ressortir les symptômes particuliers et caractéristiques des médicaments d'après leur différents rapports, j'ouvrais une voie dans le vaste champ, encore inexploré, de la combinaison."

Contrairement à l'idée que l'on se fait généralement d'un répertoire, il ne s'agit pas ici d'une simple table des matières de la matière médicale, mais d'un recueil systématique des caractéristiques générales des médicaments.

Son utilisation est, bien entendu, très spécialisée et vise à rechercher à partir de la totalité des symptômes du malade, c'est-à-dire à partir des signes caractéristiques de la maladie et du malade, le remède dont le génie recouvre cette totalité symptomatique. C'est ici qu'intervient le symptôme concomitant dont le rôle est de permettre, un peu comme dans la technique décrite par Hahnemann dans le traitement des maladies épidémiques, la différenciation du remède personnel au milieu des remèdes indiqués dans la maladie en cours.

Cette méthodologie dans le recueil des symptômes expérimentaux de la matière médicale et dans l'approche de la symptomatologie clinique, permet de comprendre et d'apprendre la matière médicale en généralisant l'une par l'autre, l'une grâce à l'autre.

Elle ouvre la porte à la réflexion sur:

- Le choix et la valorisation des symptômes.
- Leur signification.
- L'unité réactionnelle de l'individu.

Dans sa préface, Boenninghausen nous précise d'abord que cet ouvrage arrive 15 ans après la publication de son premier répertoire et il prévient ensuite :

> " Il est hors de doute qu'aucun répertoire, quelque complet qu'il soit, ne remplacera jamais l'étude consciencieuse de la matière médicale pure... "

mais constate que :

> " ... dans la pratique, il est bon qu'un résumé clair, succinct et lui offrant les principaux caractères des symptômes, vienne au secours de sa mémoire pour le mettre en état, dans tout cas de maladie concret, de choisir avec certitude, et sans perte considérable de temps, le médicament qui convient le mieux parmi les médicaments indiqués. "

Ainsi, la clef de la prescription passe-t-elle par la généralisation et il ne faut pas, bien entendu, prescrire symptôme à symptôme. Le médecin doit s'efforcer, grâce à la connaissance approfondie de la matière médicale et à l'intelligence développée par la réflexion et la pratique, de mettre en évidence la totalité symptomatique. C'est dans le but d'aider le médecin à trouver cette totalité des symptômes que Boenninghausen a conçu son Manuel.

3.5.2 Le Repertoire

Nous distinguons en général trois types de répertoires :

Dans un répertoire dit nosologique, le point de départ de la recherche est le nom de la " maladie ". A priori cette manière de faire n'est pas très homéopathique, et pourtant de très grands auteurs comme Bigel ou Hering ont écrit de semblables ouvrages de " Médecine familiale ". On pourra les consulter avec bonheur et ils nous éclaireront sur une certaine approche de la pathologie.

Les répertoires analytiques semblent plus en accord avec la méthodologie. Il s'agit le plus souvent de tables des matières de la matière médicale. Le plus célèbre a sans doute été celui de J.T. Kent qui, à l'heure actuelle reste une référence appréciée. Il faut citer chez les modernes l'énorme travail de compilation effectué par l'asbl ARCHIMEDE, d'abord, et la s.a. ARCHIBEL ensuite, et qui a donné lieu au répertoire SYNTHESIS.

" Mieux vaut une tête bien faite qu'une tête bien pleine ". C'est certainement cet adage qui a guidé Boenninghausen dans la conception de son répertoire synthétique. Contrairement aux précédents celui-ci a été construit à partir des "caractéristiques" des médicaments, en faisant appel à la clinique et en "généralisant" la notion de maladie soit artificielle, soit naturelle. Pour Boenninghausen et Hahnemann, la caractéristique d'une

maladie réside dans la partie du corps qui est affectée de telle ou telle façon et selon des conditions d'aggravation ou d'amélioration qui la "personnalisent".

La structure des répertoires est faite de recueils de rubriques composées d'un titre correspondant à un symptôme ou à une caractéristique et d'une suite de médicaments valorisés par la typographie et qui sont susceptibles de « guérir » la maladie. Cette valorisation correspond à des valeurs de 1 à 4 dans le Kent, de 1 à 5 dans Boenninghausen. Les rubriques sont regroupées par chapitres qui regroupent soit une localisation, un appareil, une fonction soit des modalités générales ou des modifications mentales. Le classement se fait par ordre alphabétique, selon les modalités, les sensations et les localisations.

3.6 INTERET DE L'ORDINATEUR

L'informatique, bien évidemment commence à jouer un rôle de premier plan parmi les outils répertoriaux car elle est parfaitement adaptée à l'homéopathie. L'ordinateur est un machine qui permet le traitement automatique de l'information et l'information est un critère qui permet de réduire un ensemble de données. Grâce à ses possibilités extraordinairement rapides de répétition, l'ordinateur va effectuer en très peu de temps des calculs et des tris qui ont demandé des heures à nos prédécesseurs. Il nous débarrasse ainsi des tâches fastidieuses.

Ses capacités de stockage des données sont énormes, Elles peuvent rendre de très grands services, notamment dans le domaine de la recherche.

Première approche

4. PREMIERE APPROCHE

4.1 ETUDE ANALYTIQUE DU REPERTOIRE

Comme il ressort de l'analyse de la méthode de Boenninghausen, il nous a semblé judicieux de regrouper le répertoire en trois grands chapitres: localisation, sensation et modalité.

4.1.1 LOCALISATION

Dans les localisations nous trouverons les chapitres suivants :

- Les symptômes mentaux: on pourrait s'étonner que ce chapitre ne couvre que huit pages du répertoire. Boenninghausen nous explique dans sa préface p. XVII du Manuel pourquoi il l'a limité à ce point:

> " La plupart des homéopathes, au début de leur carrière, ont coutume de négliger cette partie d'un tableau complet de la maladie, ou de commettre des méprises. J'ai donc cru utile de rassembler ici, sous le plus petit nombre de rubriques possible, tout ce qu'il y a de plus essentiel, afin de faciliter les recherches."

Boenninghausen a voulu ainsi attirer l'attention des débutants sur les symptômes les plus importants qu'il faut rechercher. N'oublions pas que s'il est vrai que le symptôme mental est très haut dans la valorisation encore faut-il qu'il soit sûr et caractéristique. A titre d'information nous citons quelques symptômes mentaux:

- ◊ Moral
- ◊ Intelligence
- ◊ Mémoire, etc.

- Sièges des symptômes : on l'aura compris les symptômes constituant ce chapitre font partie de notre rubrique Localisation. Pour s'en assurer extrayons ces quelques mots de la préface de l'auteur Manuel p. XVIII:

> " .... cette partie sert surtout à connaître quels médicaments agissent plus ou mois fortement sur les différents parties du corps et les différents organes. "

Dans cette partie nous trouvons tous les organes et leurs fonctions.

4.1.2 SENSATION

On trouvera ici les sensations générales (hémorragies, horreur de se laver, désir d'air libre, ...), les glandes, les os et le périoste, maladie de la peau. Les chapitres Sommeil et rêves, Circulations et fièvre sont regroupés dans les sensations.

4.1.3 MODALITES

Les modalités couvrent essentiellement le sixième chapitre du répertoire: les aggravations et améliorations

4.2 LES ETAPES PREVUES DANS CE MEMOIRE

4.2.1 LA COLLECTE DE SYMPTOMES

On ne peut pas connaître l'essence intime des maladies. Seule la symptomatologie est sûre et objective. La maladie n'est pas une entité indépendante du sujet malade. Elle n'est qu'un mot commode qui permet aux médecins de définir un ensemble de symptômes caractéristiques. Le symptôme est une " modification de l'état de santé " accessible au médecin par l'interrogatoire et l'observation clinique et para-clinique. Le recueil des symptômes se perfectionne grâce au progrès des techniques, mais il n'en est pas moins vrai que le symptôme est la seule manifestation objective et qu'il peut être soigneusement décrit.

Le travail du médecin consiste donc à interroger et examiner son malade et son entourage en fonction du motif de consultation qui ne doit jamais être perdu de vue. Cet examen et cet interrogatoire utilisent, bien entendu, toutes les ressources de la médecine moderne, mais, au-delà de cette première approche, le médecin homéopathe, qui possède une autre vision des phénomènes observés, va pouvoir mettre en évidence une grande variété de symptômes qui n'intéressent pas son confrère classique. En effet, ces symptômes ne peuvent être observés que par celui qui connaît la matière médicale homéopathique. Il est évident que l'on ne peut pas reconnaître quelque chose que l'on ne connaît pas déjà.

A cette étape le médecin disposera de quatre boîtes, et selon qu'il s'agit d'une localisation, d'une sensation, d'une modalité ou d'un concomitant, il placera son symptôme dans la boîte correspondante. A l'issue de cette phase, le médecin aura ainsi construit une répertorisation de son cas clinique. Le schéma ci-dessous illustre bien ce cas de figure.

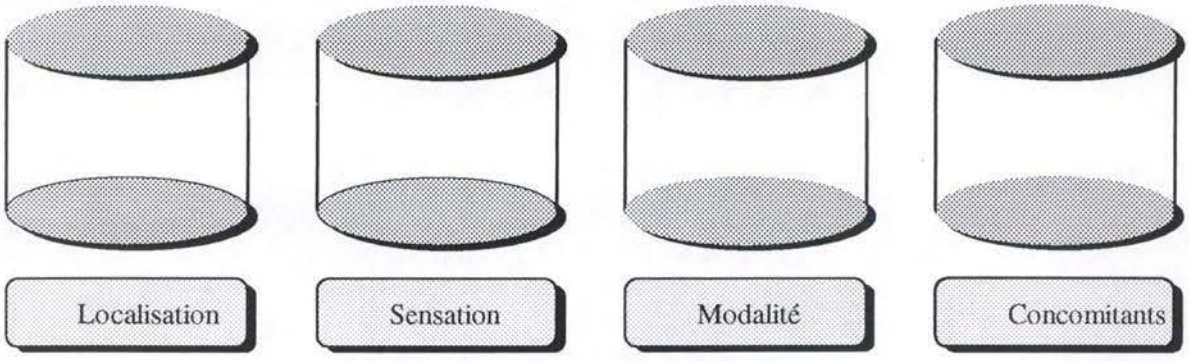


Figure 4-1 La collecte de symptômes

4.2.2 LE TABLEAU SYMPTOMATIQUE

Le médecin homéopathe devra effectuer la valorisation des symptômes. Des symptômes banals deviennent caractéristiques par leur regroupement.

Le médecin cherchera à valoriser ces symptômes selon les trois caractéristiques suivantes :

◊ Une localisation

◊ Un état morbide ou sensation

◊ Des conditions d'apparition ou d'exacerbation : des modalités

Un symptôme ne peut être retenu pour valable que s'il est ainsi valorisé.

Nous avons vu que si les symptômes groupés selon un ensemble cohérent pouvaient définir une maladie, cet ensemble est insuffisant pour englober à la fois la maladie et la manière dont le malade fait sa maladie. Dans le groupement ainsi constitué, le médecin devra inclure les symptômes concomitants qui ont pour rôle essentiel d'affiner le choix du remède.

Nous associons des numéros (1,...,8) aux différents symptômes pour ainsi traduire leur appartenance à un groupe déterminé. Ces numéros seront appelés: numéros de symptôme complet.

SYMPTOMES	NUMEROS	CARACTERISTIQUES
Esprit Absent -Distrait	❶ ❷ ❸	Loc
Irritation Inquiète	❷ ❸	Loc
Manque De Confiance	❶	Loc
Désir De Grand Air	❶	Sens
Douleur Mordante	❷ ❸	Sens
Engourdissement	-	Sens
Le Soir	❶	Mod
Périodiquement	❷	Mod
Etant Seul	❶ ❷	Mod
Voix Croassante	❶	Conc
Picotement, Fourmillement	❶ ❸	Conc
Apres Minuit	❶	Conc

Figure 4-2 Tableau symptomatique

Dans le tableau ci-dessus nous avons quatre parties. Ces quatre parties reprennent les symptômes définis dans l'étape une selon les caractéristiques permettant le regroupement. De plus nous avons une colonne `Numéros` qui sert à valoriser nos symptômes. Ainsi nous pouvons lire sur ce tableau que notre symptôme `Esprit Absent -Distrait` qui est une localisation entre en considération dans nos symptômes complets ❶, ❷, ❸. En outre le symptôme `Engourdissement` dans sensation n'a pas été valorisé.

4.2.3 VERIFICATION DU CAS

Cette étape constitue un outil intéressant pour le médecin. Au risque de se noyer sous le flot d'information, il s'agira ici de permettre au praticien de voir parmi ses symptômes complets ceux qui ne le sont pas. Pour rappel un symptôme est dit complet et ne peut être retenu comme valable que s'il est valorisé selon les trois caractéristiques:

◊ Localisation

◊ Etat morbide et sensation

◊ Modalités

Le tout est de signaler au médecin homéopathe que, pour tel numéro de symptôme complet en état incomplet, il manque telle ou telle caractéristique. L'étude du tableau symptomatique défini précédemment donnerait le résultat suivant

	SYMPTOM COMPLET ❶
Localisation	Esprit Absent -Distrait Manque De Confiance
Sensation	Désir De Grand Air
Modalité	Périodiquement Etant Seul
Concomitants	Voix Croassante Picotement, Fourmillement Apres Minuit

Figure 4-3 Tableau du symptôme complet ❶

Le médecin qui verrait un tel tableau illustré par la Figure 4-2 serait sans doute satisfait de la forme, mais peut-être moins du contenu. Ce cas de figure est en état complet. Les trois caractéristiques déterminantes sont présentes. De plus on peut y noter la présence de symptômes concomitants qui permettent la "personnalisation", l'individualisation du cas.

	SYMPTOM COMPLET ❸
Localisation	Esprit Absent -Distrait Irritation Inquiète
Sensation	Douleur Mordante
Modalité	?
Concomitants	Picotement, Fourmillement

Figure 4-4 Tableau du symptôme complet ❸

Ce cas de figure est intéressant à signaler au médecin homéopathe qui souhaiterait certainement compléter son symptôme complet. Dans le cas présent, ce sont les modalités qui font défaut. De deux choses l'une, soit durant la phase de la collecte des symptômes la boîte des modalités est restée vide, soit durant la phase de valorisation notre médecin homéopathe a omis de valoriser les modalités entrant en compte dans ce symptôme complet.

4.2.4 ANALYSE DU CAS

L'homéopathe adepte de la méthode de Boenninghausen va dresser le tableau symptomatique du patient où l'on trouvera les caractéristiques suivantes: une localisation, une sensation, une modalité, et des symptômes concomitants. Il va oeuvrer afin de réduire l'ensemble symptomatique à un syndrome minimum de valeur maximale.

Il se servira d'un tableau à double entrée: pour chaque symptôme s_i , le remède r_j figurera avec la valeur 5, 4, 3, 2, 1, appelée degré, et suggérant une charge probabiliste décroissante.

Le remède qui a le plus de chances de guérir l'affection, devrait couvrir l'ensemble symptomatique et être affecté de la plus grande charge en degré.

Dans le cas présent, nous choisirons les symptômes suivants que nous regroupons de la façon suivante:

◊ Localisation: `Throat - Pain - Warm drinks`

◊ Sensation: `Mind - Loquacity - Heat - During`

◊ Modalité: `Throat - Choking - Agg`

◊ Concomitant: `Mind - Jealousy`

	Loc	Sens	Mod	Conc	Total
Agar	-	-	1	-	1
Ambr	-	-	1	-	1
Apis	2	-	2	2	6
Coff	1	1	-	-	2
Lach	3	2	3	3	11
Stram	2	2	-	-	4

Table 1 Tableau analyse

L'examen de ce tableau montre que:

◊ Lachesis couvre la totalité des symptômes avec un poids de 11 degrés.

◊ Apis couvre 3 symptômes sur 4, avec un poids de 6 degrés.

◊ Stromonium couvre 2 symptômes pour 4 degrés.

◊ Coffea couvre 2 symptômes pour 2 degrés.

◊ Lachesis ou Apis apparaissent comme les remèdes les plus indiqués.

Possibilités dans RADAR

5. LES POSSIBILITES DANS RADAR

L'objectif de ce mémoire est d'élargir le système d'aide au diagnostic proposé dans RADAR et développé d'abord par l'asbl ARCHIMEDE puis par la sa ARCHIBEL. Ainsi RADAR devrait pouvoir offrir une autre approche dans le choix du remède, avec notamment la méthode de Boenninghausen. Cette nouvelle possibilité ne sera pas une entité isolée, mais s'inscrira plutôt dans le système informatique existant. Le tout, bien sûr, devra être un ensemble cohérent et harmonique. Dans ce travail on trouvera pas mal de procédures reprises dans RADAR ou réadaptées selon les besoins, pour la réalisation de ce projet. Toujours dans un souci d'harmonie et de continuité dans nos spécifications, nous expliquerons si nécessaire, mais brièvement certaines procédures dans RADAR. Les sources de RADAR étant des données privées, le lecteur intéressé devra s'adresser à la sa ARCHIBEL pour plus d'informations.

5.1 LE CHOIX DES OUTILS

Dans ce mémoire, nous utilisons le langage C. Ce langage a été adopté par ARCHIBEL, pour l'élaboration de RADAR.

Le langage C est caractérisé par sa souplesse et sa complexité. Il n'impose pas vraiment de contraintes majeures. Cette souplesse et cette complexité fait de lui un outil puissant, mais demande une certaine expérience dans les fondements mêmes de la programmation.

Le langage C est un langage de programmation conçu pour de multiples applications. Il est souvent associé au système Unix, sur lequel il a été développé, car ce système est entièrement écrit en C, ainsi que la majorité des logiciels qui tournent sous Unix. Néanmoins, ce langage n'est lié à aucune machine, ni à aucun système d'exploitation.

Ceci constitue une référence valable pour le choix judicieux du langage C. Car nous pouvons signaler la robustesse et la stabilité du système Unix.

Le langage C fournit les structures de contrôle nécessaires à l'élaboration de programmes structurés : le regroupement des instructions, les prises de décisions (if, else), le choix d'un cas parmi un ensemble de possibilités (switch), les boucles avec le test de sortie au début (while, for) ou à la fin (do), et la sortie de boucle anticipée (break).

Les fonctions peuvent retourner n'importe quel type : les types de base, des structures, ou des pointeurs. Elle peuvent aussi figurer dans des fichiers sources distincts qui sont compilés séparément. Une étape de précompilation s'occupe de substituer les macro-instructons dans le texte du programme, d'inclure d'autres fichiers sources.

Le langage C est un langage que l'on peut qualifier de "bas niveau". Cette appellation n'est pas péjorative; elle signifie simplement que le C manipule les mêmes sortes d'objets que la plupart des ordinateurs à savoir des caractères, des nombres, et des adresses. On peut combiner ces objets entre eux et les manipuler grâce aux opérateurs arithmétiques et logiques que les machines réellement. Le langage C ne comporte aucune opération qui traite

directement des objets composés, tels que les chaînes de caractères, les listes ou les tableaux.

La norme ANSI apporte au langage une seconde contribution de taille : elle définit une bibliothèque d'accompagnement. Celle-ci spécifie des fonctions qui permettent d'accéder au système d'exploitation (par exemple, pour lire et écrire des fichiers), d'allouer de la mémoire, de manipuler des chaînes de caractères. Elle garantit un fonctionnement compatible aux programmes qui l'utilisent pour dialoguer avec le système d'exploitation sous lequel ils tournent.

Un des atouts du langage est la possibilité de pouvoir allouer dynamiquement de l'espace mémoire. Dans notre travail toutes les allocations se font d'une manière dynamique. C'est une tâche pas facile à réaliser, mais qui est d'une importance capitale. En effet quand on développe une application d'une taille relativement considérable, un tel choix se justifie. Contrairement à une allocation dite statique, l'allocation dynamique permet au programmeur de gérer son espace mémoire. Ainsi il pourra décider au cours du déroulement de son programme de charger ou de décharger telles ou telles informations en mémoire suivant que la donnée est nécessaire ou pas au moment de l'exécution du programme.

Pour ce qui est des outils nous servant dans la conception de l'interface, nous utilisons le XVT design. XVT est entièrement écrit en C, d'où sa stabilité et sa portabilité dans un programme conçu en C. RADAR ne tourne pas uniquement sous PC ou sous WINDOWS, il est aussi destiné au produit MAC. Comme l'illustre la *figure 4-1*, XVT a l'avantage de générer un code portable sous WINDOWS ou MAC entre autre.

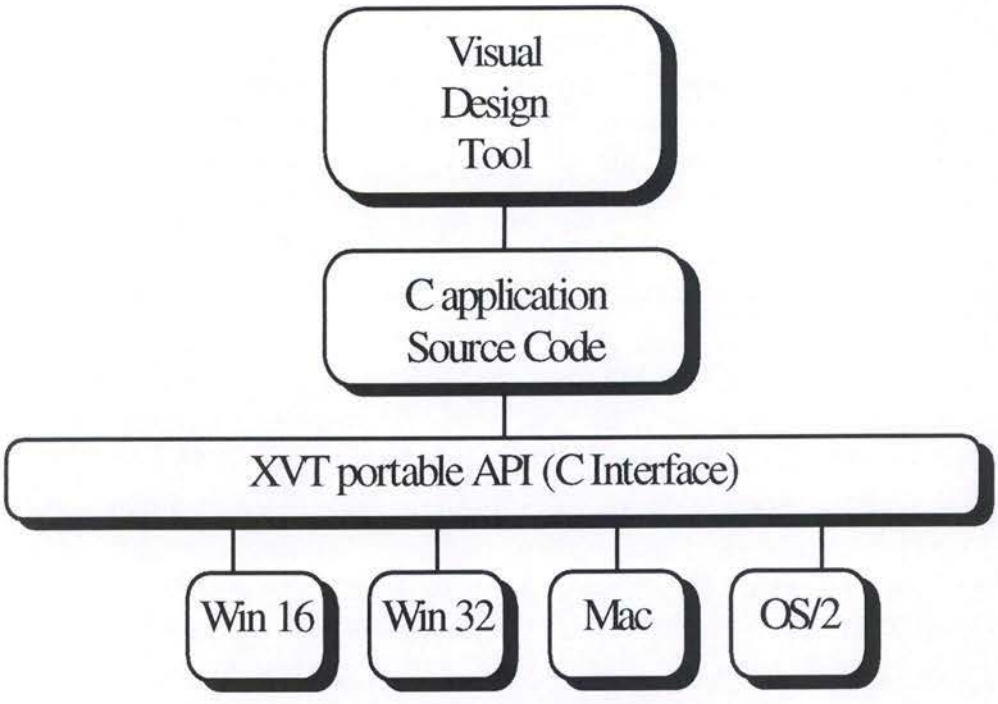


Figure 5-1 XVT Potability Toolkits

Nous résumons dans la *figure 4-2* les différentes étapes, et enchaînements d'une application développée sous XVT :

- ◊ Les ressources URL (Universal Resources Langage) sont compilées sous la bonne plate-forme
- ◊ Le Project file, et le code source du programme sont associés pour être compilés. On associera code application avec GUI (Graphical User Interface), c'est-à-dire comment et quand les objets communiquent les uns avec les autres.
- ◊ Au moment de l'édition des liens on fait appel à la librairie XVT, et la librairie soutenant notre GUI (Graphical User Interface). Le tout produit un code exécutable.

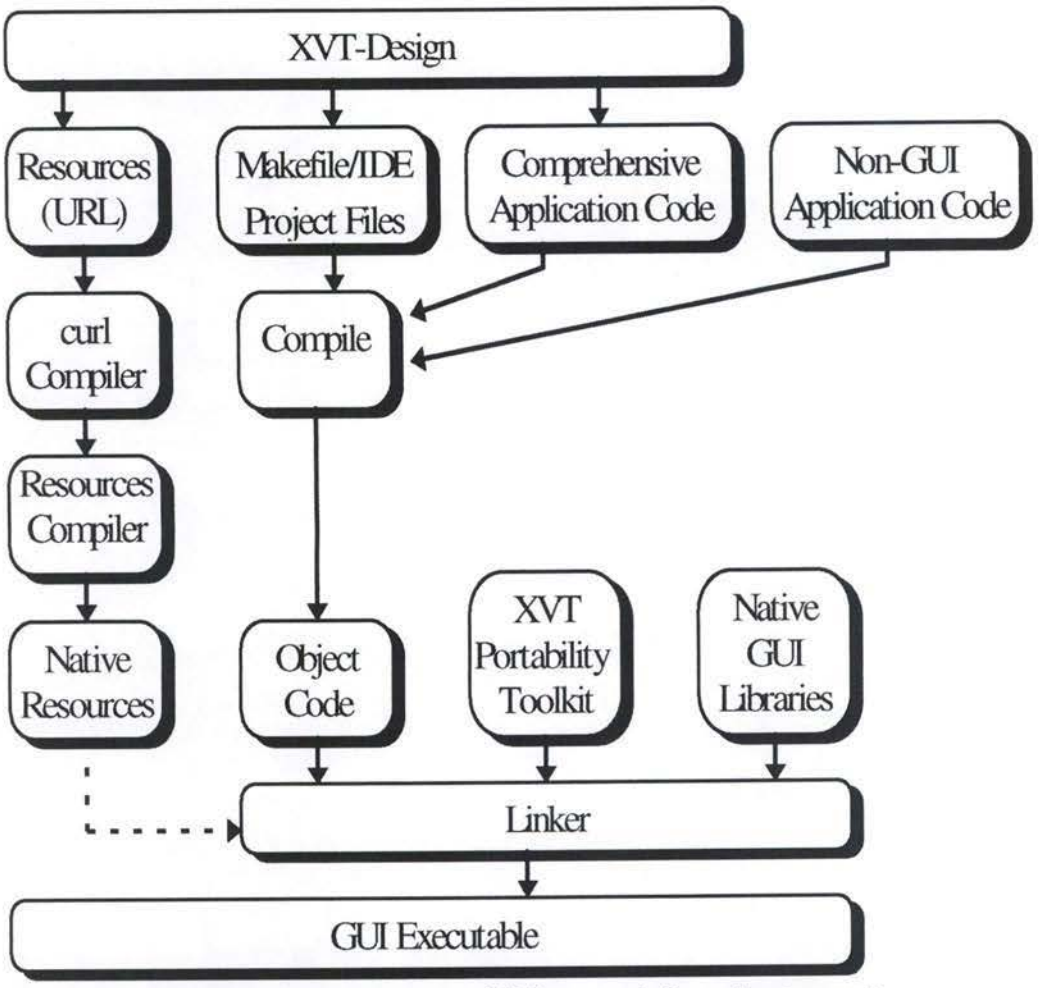


Figure 5-2 Important steps of "C" cross-platform development

Ainsi, par exemple pour créer un fenêtre, on appelle la fonction sous XVT qui crée une fenêtre, sans soucier de la bonne plate-forme. XVT se chargera plus tard de gérer les appels WINDOWS, MAC, ou autres.

5.2 ARCHITECTURE DE RADAR

La *figure 4-3* montre la relation entre le fichier Patient et le module de répertorisation. On trouve les étapes nécessaires pour inscrire les données d'un patient, et de sa

consultation. On peut rattacher une répertorisation à un patient si nécessaire. La répertorisation offre plusieurs possibilités:

- L'utilisation d'un ou de plusieurs répertoires dans lesquels on peut chercher des symptômes.
- Extraction des symptômes trouvés dans une liste.
- Modifier des symptômes, rendre certains d'entre eux plus importants que d'autres :
  - ◊ Exclure certains degrés de remèdes.
  - ◊ Grouper ou effacer des symptômes dans une liste,...
- Analyser votre cas, selon différentes manières :
  - ◊ Somme des symptômes.
  - ◊ Somme des degrés, ...

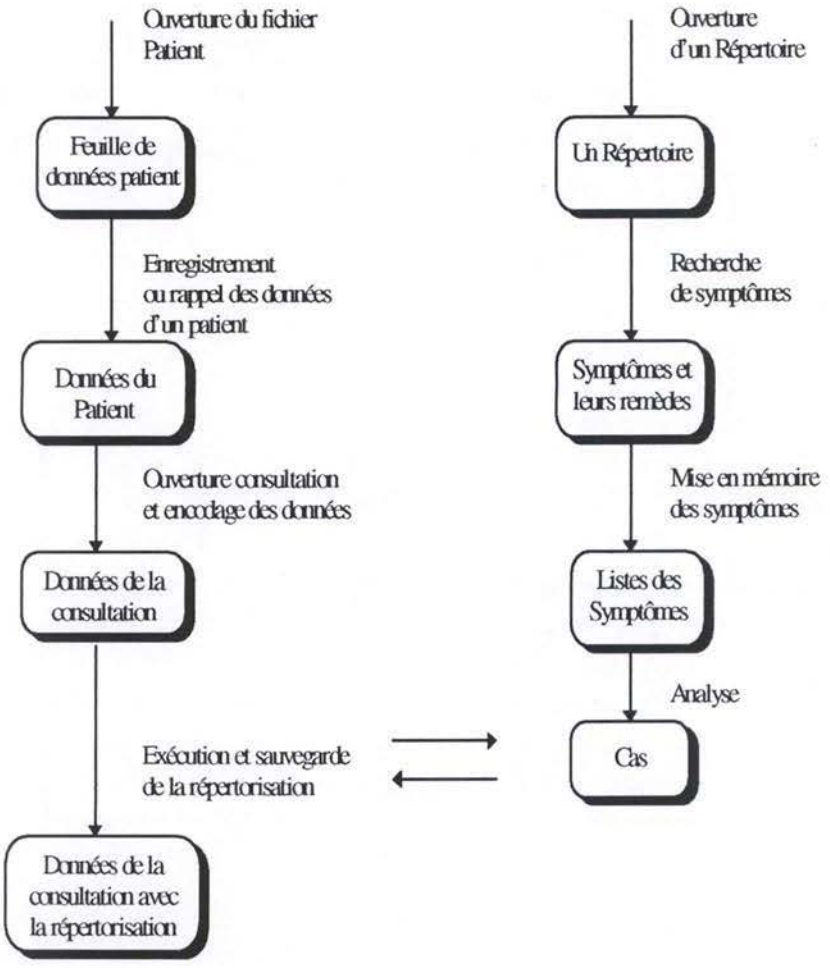


Figure 5-3 Architecture de RADAR

5.3 LA SYNTAXE DU REPERTOIRE

Signalons d'abord le travail remarquable de l'asbl Archimede. La compilation du répertoire de Boenninghausen, est un fichier texte MS-DOS, moyennant une syntaxe singulière.

Le nom de la rubrique est toujours précédé du caractère & suivi d'un chiffre qui correspond au niveau d'imbrication. On lira dans l'exemple ci-dessous que `Margins Of Hair` est un sous-niveau de `Mind` qui est lui-même un sous-niveau de `Localisation`. Le nom de la rubrique qui est ici en anglais, est suivi du caractère fin de ligne et de sa traduction en français. Pour passer du nom de la rubrique et de sa traduction à la liste des remèdes, nous ajoutons vers la fin les caractères deux points et fin de ligne. Notons que cette dernière règle reste valable même si la liste des remèdes est inexistante pour une rubrique donnée. La liste des remèdes finit toujours par un "deux points" suivi du caractère fin de ligne.

Les différents remèdes sont séparés par une virgule. Il est souhaitable d'en avoir au maximum quatre sur une ligne. Le nom du remède est suivi d'un point, d'un chiffre qui correspond au degré du remède, des lettres `BXYZ` entre crochets. Par convention nous avons choisis ces lettres pour ainsi signifier l'auteur Boenninghausen.

```
&1Localisation
Localisation:
&2Mind
Mental:
&3Margins Of Hair
Racine Des Cheveux:
calc.2[BXYZ],nat-m.4[BXYZ],petr.3[BXYZ],sep.4[BXYZ],
tell.3[BXYZ]:
```

Un extrait de la base de données

Ce travail est indispensable pour la suite. Dans la conception du logiciel RADAR, l'équipe d'Archibel a mis au point des procédures permettant la traduction et l'implémentation des différents répertoires dans leur système informatique.

5.4 OUVRIR UN REPERTOIRE

Ainsi dans RADAR pour ouvrir le répertoire de Boenninghausen, comme tout autre répertoire disponible, il suffit d'aller dans le menu **Fichier**, et choisir **Ouvrir Répertoire**. Dans la *figure 5-4*, la fenêtre des répertoires disponibles est affichée. Le répertoire de Boenninghausen est en surbrillance; pour l'ouvrir cliquez sur **OK**.

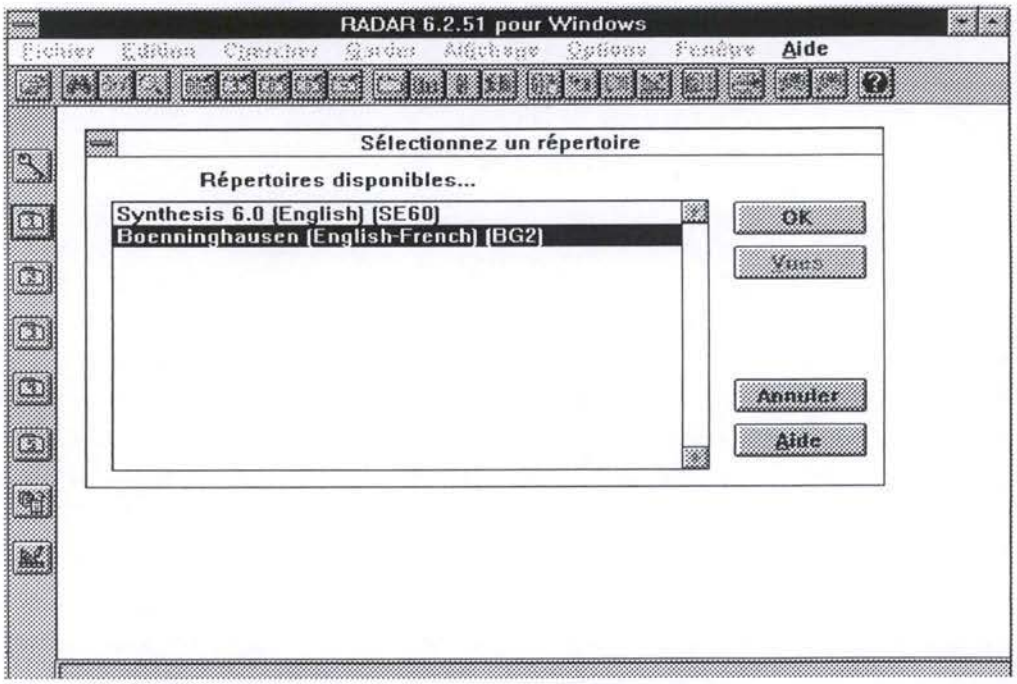


Figure 5-4 Fenêtre Répertoires disponibles

Remarque: on sait accéder à un répertoire à partir de la barre d'outils ou du clavier. Pour ce faire consulter l'aide en ligne (rechercher " Ouvrir un répertoire " dans le mode de recherche).

5.5 LA RECHERCHE DANS RADAR

RADAR offre deux possibilités pour rechercher des symptômes dans le répertoire. La première est proche de celle utilisée pour la recherche dans un répertoire imprimé. Vous trouvez le chapitre. Ensuite, vous cherchez la rubrique qui vous intéresse dans ce chapitre. Cette façon de faire est appelée *Trouver un symptôme*.

La seconde option est la facilité de *recherche*. Ceci est un instrument très puissant qui vous permet de rechercher des mots dans le répertoire, d'extraire des remèdes du répertoire et d'exécuter une recherche en utilisant une combinaison de critères. En fait, vous devez demandez au système de chercher quelque chose de précis qui répond à vos critères de recherche. Pour limiter la recherche, vous pouvez demander au système de rechercher dans un chapitre spécifique et/ou de considérer certaines modalités.

5.5.1 TROUVER UN SYMPTOME

C'est une manière rapide de trouver des symptômes dans le répertoire Avec cette procédure, vous pouvez facilement choisir un chapitre du répertoire et voir ses rubriques. Pour rappel il faudrait d'abord ouvrir un répertoire pour effectuer une recherche de symptômes.

Pour trouver un symptôme:

◊ Cliquez sur le bouton ou choisissez **Trouver symptôme** dans le menu **Chercher**. La fenêtre *Sélection du chapitre* s'ouvre.

◊ Dans la fenêtre *Sélection du chapitre*, cliquez sur l'icône représentant le chapitre choisi. La fenêtre *Sélection du symptôme* s'ouvre et montre le titre du chapitre dans la partie haute de l'écran (voir *figure 5-5* La fenêtre Sélection du symptôme)

◊ Dans la fenêtre *Trouver: sélectionnez symptôme*, vous verrez une barre contenant des boutons caractères (par ordre alphabétique).

En cliquant sur la première lettre d'une rubrique, RADAR affiche la rubrique désirée dans la partie basse de l'écran. Remarquons que les symptômes figurant à cet endroit de l'écran, montrent les rubriques commençant par la lettre choisie.

Cliquez sur la touche Retour arrière pour corriger les erreurs éventuelles.

◊ Dès que le symptôme choisi apparaît dans le bas de l'écran, cliquez une fois dessus. Le symptôme apparaîtra à la suite du titre de chapitre qui se trouve dans la partie haute de la fenêtre. S'il y a une sous-rubrique pour le symptôme sur lequel vous avez déjà cliqué, elle sera automatiquement visible dans le bas de l'écran. Cliquez sur la sous-rubrique afin de la ramener vers le haut de la fenêtre.

◊ Choisissez **OK** pour retourner sur le symptôme choisi dans la fenêtre du répertoire.

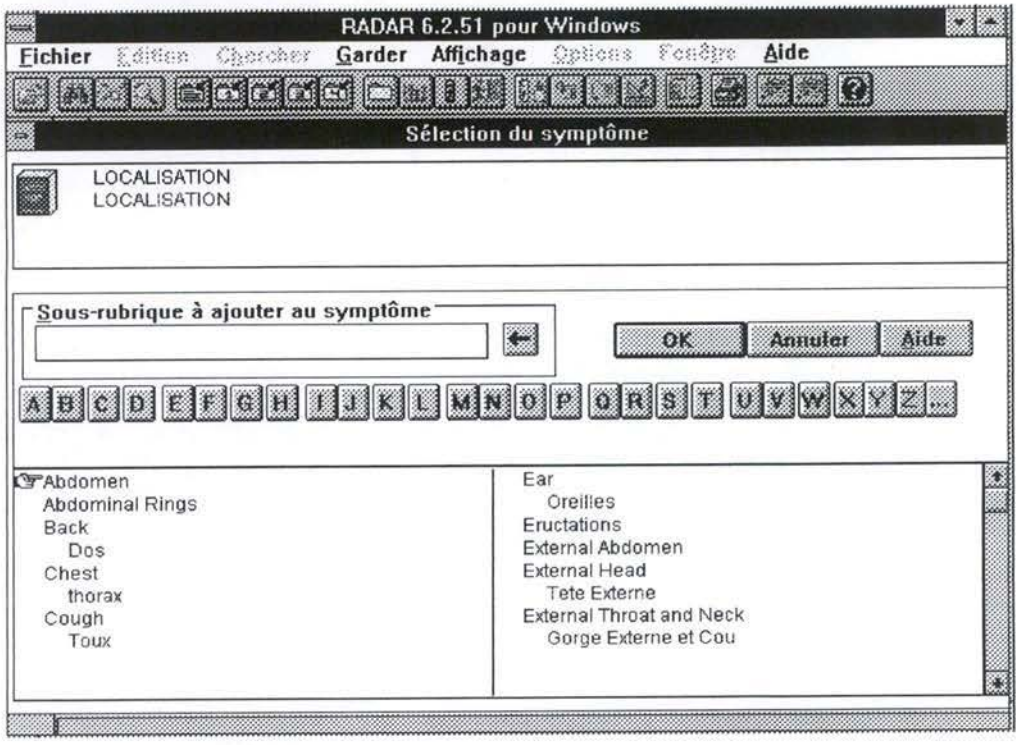


Figure 5.5 La fenêtre Sélection du symptôme

Note sur l'accès rapide: Dans la fenêtre de répertoire, tapez juste les premières lettres du chapitre que vous recherchez. Remarquons qu'une fenêtre spéciale s'ouvre automatiquement, affichant les caractères que vous tapez. Appuyez ensuite sur ENTER pour trouver la fenêtre *sélection du symptôme*.

5.5.2 TROUVER DES MOTS SPECIFIQUES DANS LE REPERTOIRE

RADAR vous permet de chercher dans le répertoire des mots spécifiques. Vous pouvez aussi rechercher une combinaison de mots

Pour trouver des mots spécifiques dans le répertoire:

- ◊ A partir de la fenêtre du répertoire, choisissez Recherche sur mots dans le menu Chercher ou appuyez sur le bouton dans la barre d'outils, pour ouvrir la fenêtre Recherche sur mot.
- ◊ Dans la barre de caractères, cliquez sur la première lettre du mot que vous chercher, vous pouvez aussi tapez le premiers caractères. Dès que le mot apparaît dans la partie basse de l'écran, cliquez dessus et il apparaîtra dans la partie supérieure droite de la fenêtre. Le mot sera affiché entre "guillemets" précédé de "mot".
- ◊ Si vous désirez ajouter d'autres mots, cliquez sur la première lettre du second mot. Dès que le nouveau mot est affiché dans la partie basse de l'écran, il vous suffit de cliquer dessus afin de l'amener vers la partie supérieure droite. Dans l'exemple ci-après, l'étape de la recherche des mots sera, par exemple le mot: ongle ET le mot: mordant.
- ◊ Cliquez sur **Chercher** pour démarrer la recherche. Cliquez sur **Annuler** pour fermer la fenêtre sans avoir à effectuer de recherche. Si vous voulez effacer la partie supérieure droite de l'écran, cliquez sur **Effacer**.

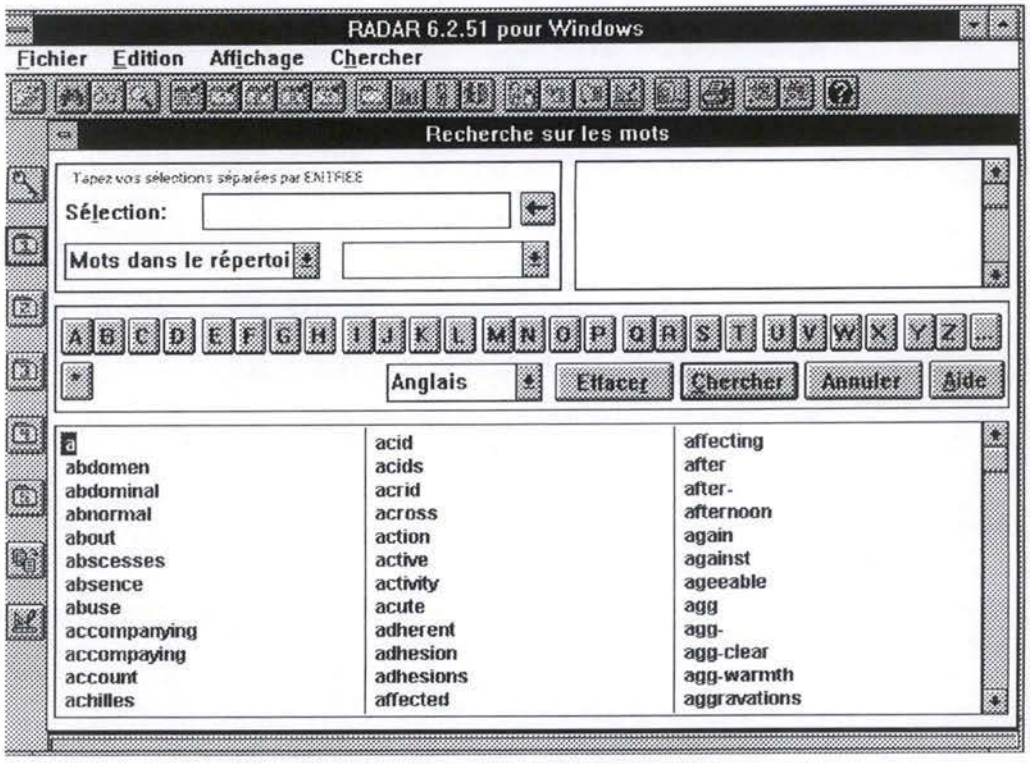


Figure 5-6 La fenêtre de recherche sur les mots

Notes:

- Si vous ne trouvez pas le mot comme une *racine*[8] ou une *branche*[9], cliquez sur le bouton adjacent. Un sous-menu s'ouvre, dans lequel vous pouvez choisir mots dans le répertoire. Vous pouvez aussi tapez **SHIFT + F1** pour chercher *mots dans le répertoire.*
- Si vous désirez chercher plus d'un mot sans utiliser l'opérateur **ET**, cliquez sur le bouton adjacent ou le **TAB** jusqu'aux opérateurs et choisissez **OU, ET NON** (utilisez les touches fléchées).

5.5.3 EXTRACTION DE REMEDES DU REPERTOIRE

Il existe une large variété d'options permettant l'extraction de remèdes à partir du répertoire. Vous pouvez extraire juste un remède ou une combinaison de remèdes. Vous pouvez aussi définir des critères pour l'extraction comme la taille des symptômes à inclure dans l'extraction. Vous pouvez aussi définir la partie dans le répertoire dans laquelle l'extraction aura lieu.

Extraire des remèdes du répertoires:

◊ Choisissez **Extraction comparative** dans le menu **Chercher** pour ouvrir la fenêtre *Extraction simple.*

◊ Cliquez sur le bouton affichant **1** dans la partie supérieure gauche de la fenêtre pour ouvrir la fenêtre de *Sélection des remèdes.*

◊ Cliquez sur la première lettre du remède, figurant dans la barre de caractères de la fenêtre de sélection de remèdes, que vous désirez extraire. Dès que vous visualisez ce remède dans la partie inférieure de l'écran, cliquez sur le bouton **OK**. Remarquez que le remède choisi est affiché juste derrière le premier bouton dans la fenêtre *Extraction comparative.*

◊ Cliquez sur **Extraire** pour démarrer l'extraction. Cliquez sur **Annuler** pour fermer la fenêtre sans avoir l'extraction. Cliquez sur **Effacer** pour supprimer la partie supérieure droite de l'écran et recommencer une nouvelle extraction

[8] Une racine correspond à une rubridue de niveau supérieur
[9] Une branche est un sous-niveau d'une rubrique

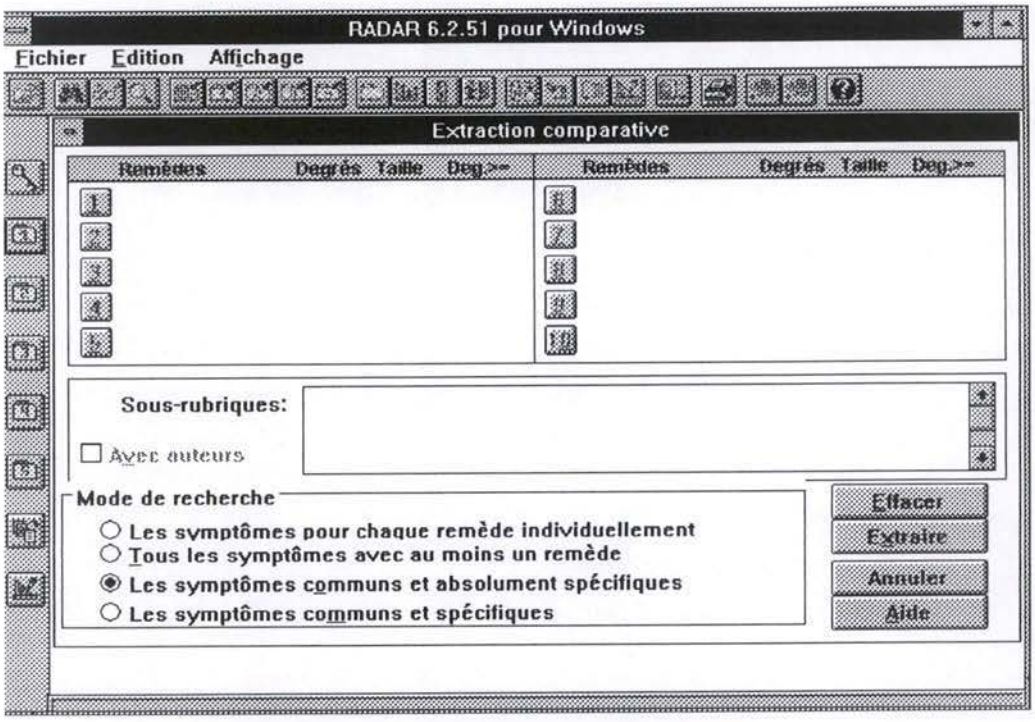


Figure 5-7 Fenêtre de l'extraction comparative

Organisation et Présentation de l'Application

6. ORGANISATION ET PRESENTATION DE L'APPLICATION

6.1 INTRODUCTION

Jusqu'à présent, après avoir présenté la méthode de Boenninghausen, nous avons décrit dans une première approche, une idée assez intuitive de l'organisation de l'application, et nous avons présenté les outils existants dans RADAR notamment pour la recherche des symptômes.

Le contenu du chapitre présent est, quant à lui, d'une portée plus proche de l'application (et donc plus technique). Il décrit, à la fois, les principales structures de données, et algorithmes entrant dans notre application. Ce chapitre nous donne aussi une idée assez large des possibilités de l'application.

6.2 ARCHITECTURE GENERALE

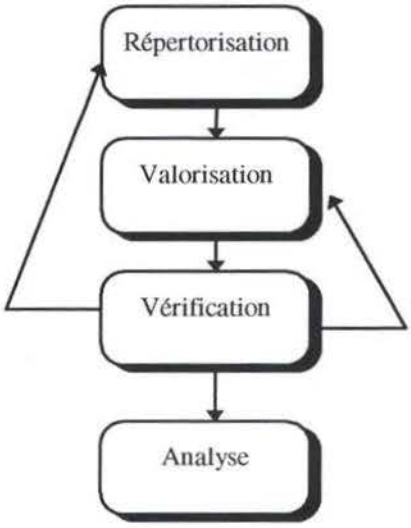


Figure 6-1 Architecture générale

La *figure 5-1* traduit l'architecture générale de notre application. Le module répertorisation concerne essentiellement la mise en mémoire des symptômes. Ce module est suivi d'une phase de valorisation. Le module valorisation permet de dresser le tableau symptomatique du patient. Le module vérification permet de vérifier l'état complet ou incomplet des différents cas enregistrés. Si le cas est incomplet, le médecin homéopathe peut revenir à la phase de répertorisation ajouter des symptômes, ou revenir à la phase de valorisation associer à son numéro la caractéristique faisant défaut. Le module analyse fournit le ou les remèdes susceptibles de guérir le patient.

6.2.1 MODULE REPERTORISATION

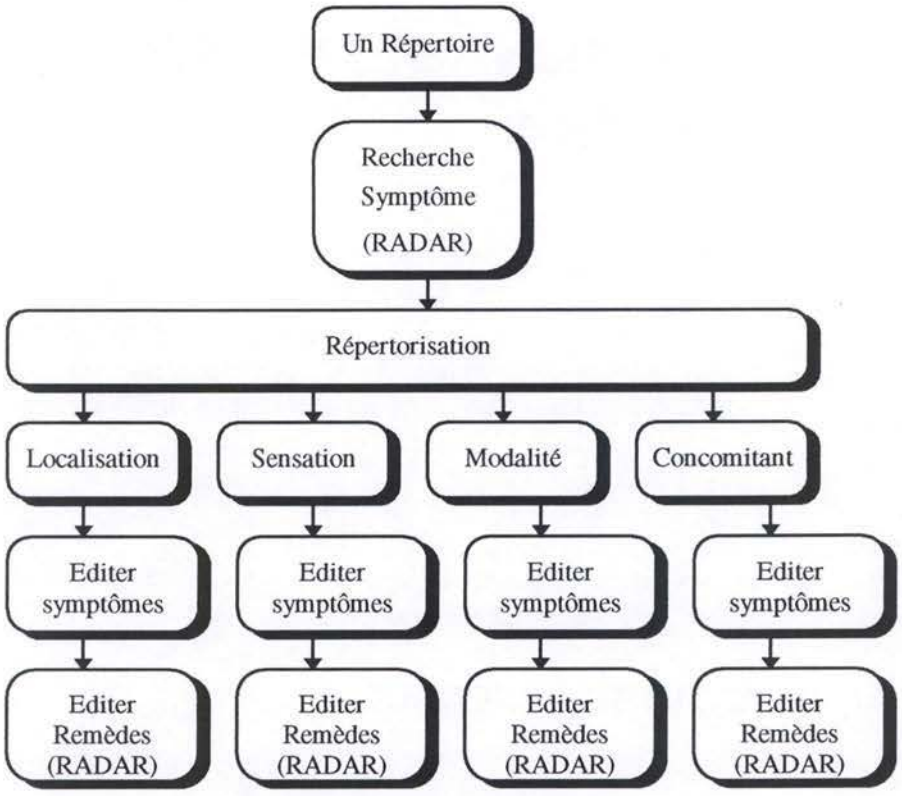


Figure 6-2 Module répertorisation

Comme dit précédemment et l'illustre la *figure 5-2*, le module répertorisation sert à enregistrer les symptômes présentés par le patient. Ainsi selon qu'il s'agit d'une localisation, d'une sensation, d'une modalité, ou d'un symptôme concomitant, le symptôme sera enregistrer dans la bonne liste.

Ce module permettra aussi d'éditer les symptômes enregistrés dans une des quatre caractéristiques. Ce module communique avec RADAR. En effet, il sera possible d'éditer les remèdes associés à un symptôme par l'appel d'une fonction dans RADAR. Un autre point de communication avec RADAR, sera la phase de recherche de symptômes dans un répertoire donné.

6.2.2 MODULE VALORISATION

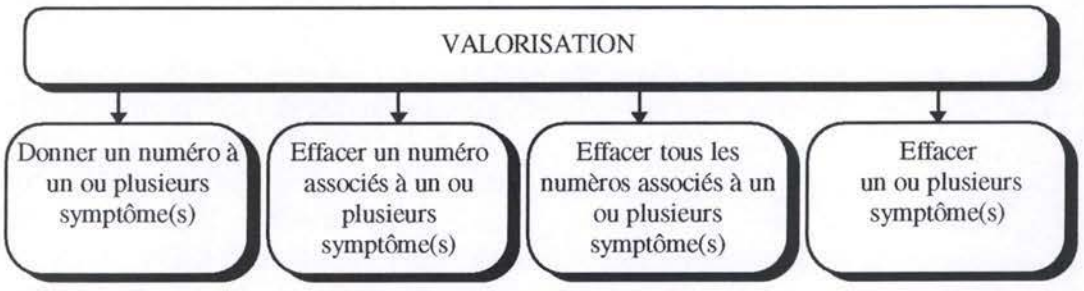


Figure 6-3 Module valorisation

Dans ce module *figure 5-3*, on prévoira les fonctionnalisés suivantes:

- Associer un numéro de symptôme complet à un symptôme, ou à un groupe de symptômes sélectionnés
- Effacer un numéro de symptôme complet associer à un symptôme, ou à un groupe de symptômes sélectionnés.
- Effacer un symptôme, ou groupe de symptômes sélectionnés. Cette dernière fonctionnalité se fera par l'appel d'une procédure dans RADAR.

Nous avons illustré par la figure 5-4 les enchaînements qui suivent la phase de valorisation. En effet on pourra éditer un symptôme complet, ou le cas complet constitué de un ou plusieurs symptômes complets.

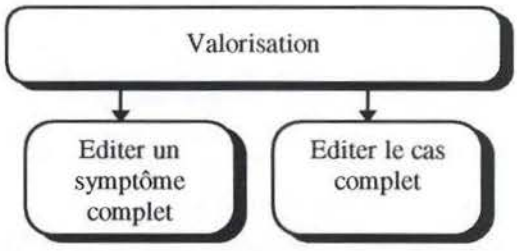


Figure 6-4 Edition du tableau symtomatique

6.2.3 MODULE ANALYSE

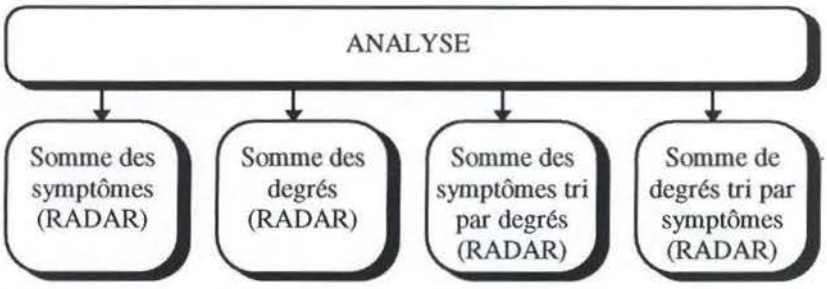


Figure 6-5 Analyse et options

Notre analyse tiendra compte des options disponibles dans RADAR. Ainsi on pourra faire l'analyse selon les critères suivants:

- Somme des symptômes: on compte le nombre de fois que le remède apparaît dans les différents symptômes.
- Somme de degrés: on totalise les degrés
- Somme des symptômes et tri sur les degrés: les remèdes apparaîtront selon la somme des symptômes et trier sur leurs poids.
- Sommes des degrés et tri sur le symptômes: les remèdes apparaîtront selon leur poids et trier sur la somme des symptômes.

6.2.4 EDITION DU CAS CLINIQUE

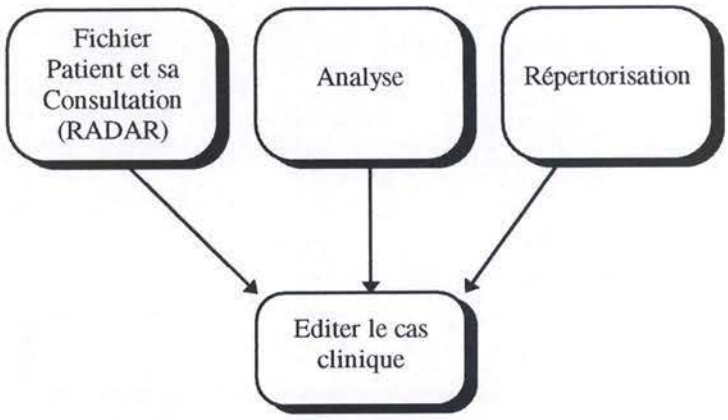


Figure 6-6 Edition du cas clinique

Cette dernière étape servira de résumé au cas. Ainsi on éditera les donnés du patients (RADAR gère un fichier patient, où on trouve les cordonnées du patient, et les données de la consultation), sa répertorisation (les symptômes de la maladie, et les symptômes du malade), et les résultats de l'analyse.

6.3 LA STRUCTURE D'UN SYMPTOME

L'information nécessaire, et qui demeure au centre de notre application, est sans aucun doute le symptôme. Au-delà du simple fait que le symptôme a un nom, on trouve tout un ensemble de remèdes associés au symptôme. La structure traduisant un symptôme dans notre cas, est la même que celle définie dans RADAR. Ceci est plus qu'indispensable pour la portabilité et l'appel de certaines procédures dans RADAR. Un tel choix trouve sa justification dans le chapitre précédent où nous avons présenté les possibilités dans RADAR avec notamment les outils de recherche de symptômes.

Cependant cette structure dans RADAR n'est pas tout à fait conforme avec notre application. Nous trouvons des champs qui n'entrent pas en considération dans notre application, ou parfois qui n'ont aucun sens avec la méthode. Le champ `Intensity` dans cette structure est important dans la méthode proposée par Kent. Selon l'auteur, lors de l'examen clinique, plus un symptôme est frappant et caractéristique, plus il a de l'importance. La valeur de l'intensité d'un symptôme varie entre 1 et 4. Mais chez Boenninghausen, il n'y a pas un symptôme qui prévale sur un autre. Des symptômes banals deviennent caractéristiques par leur regroupement. Un tel champ ne présente donc guère d'importance, mais n'est pas non plus en contradiction avec la méthodologie. Toujours dans notre souci de conserver cette structure, ce champ a été maintenu et sa valeur est fixée à 1 pour tout symptôme, 1 étant l'élément neutre de la multiplication. Tous les symptômes seront ainsi considérés avec le même poids.

Ce qui fait défaut dans cette structure et qui a une importance capitale pour notre application, est un numéro de symptôme complet. Un tel champ permettrait le regroupement et la valorisation de nos symptômes. Cependant un compromis a été établi, les champs `Degrees`, `Qualif`, serviront à dresser notre tableau symptomatique.

Le champ `SptTextLg1` sert à afficher notre symptôme dans la première langue.

Le champ `SptTextLg2` sert à afficher notre symptôme dans la deuxième langue.

```
   /*---- TYPEDEF SECTION --------------------------*/
typedef struct {
    unsigned char    *SptTextLg1;
    unsigned char    *SptTextLg2;
    char             Intensity;
    unsigned char    Group;
    REMED_LIST       *RmdList;  /* Pointer to a table of Remedy List */
    unsigned char    Degrees;
    unsigned char    Qualif;
    unsigned short   NbRemedies;
    char             Langue[2];
    unsigned char    Origin;
    unsigned short   View;
} SYMPTOM_IN_MEMORY;
typedef struct {
    unsigned short   RemedId;
    unsigned char    Degree;
    unsigned short   AuthorId;
} REMED_LIST;
```

6.4 LA COLLECTE DE SYMPTOMES

Une fois que le symptôme souhaité est trouvé, le médecin s'il le souhaite pourra garder son symptôme. Dans une approche intuitive nous avons proposé quatre boîtes correspondant aux quatre caractéristiques de regroupement. Plus concrètement, ces quatre boîtes sont représentées par des "clipboards":

- ◊  pour désigner le clipboard des localisations.
- ◊  pour désigner le clipboard de sensations.
- ◊  pour désigner le clipboard de modalités.
- ◊  pour désigner le clipboard des concomitants.

A partir de la fenêtre du répertoire de Boenninghausen, pour garder un symptôme, cliquez sur le symptôme désiré et faites un "drag and drop" vers un des "clipboards" cités au dessus.

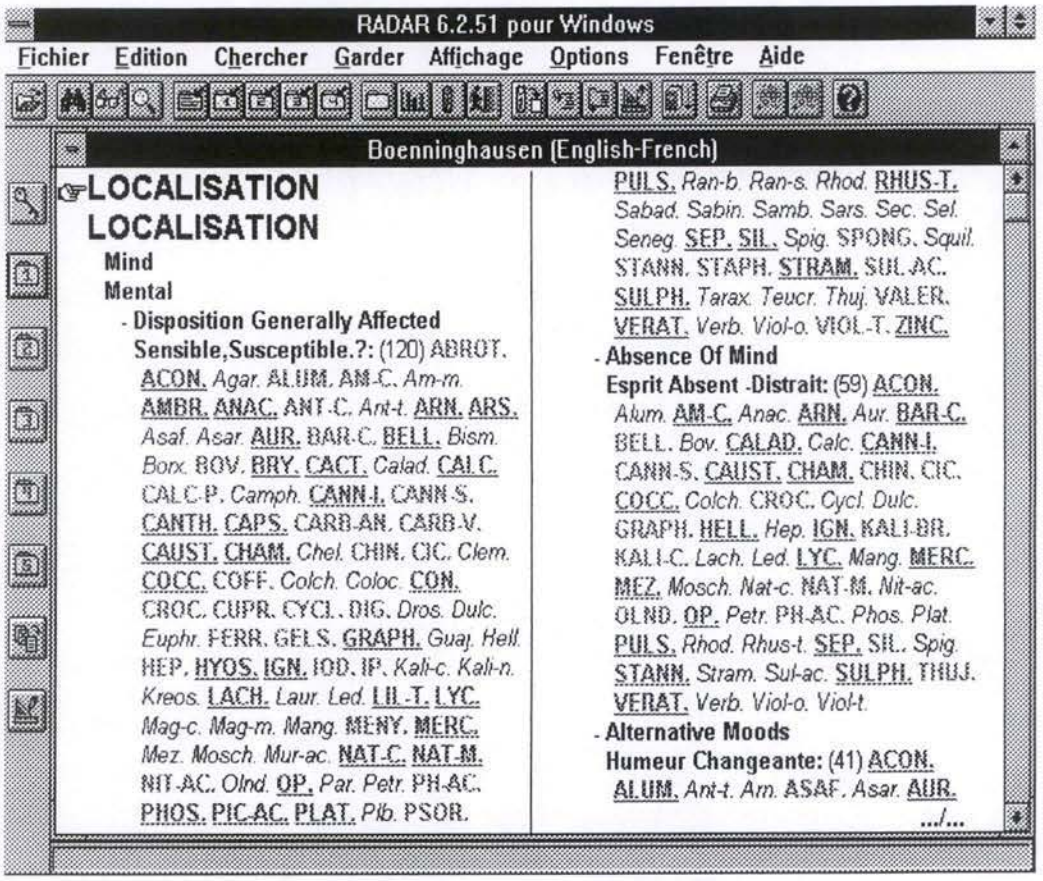


Figure 6-7 Fenêtre du répertoire de Boenninghausen.

L'ensemble défini par ces quatre "clipboards", sera implémenté sous forme de tableau `SYMPTOM_CLIPBOARD`. Ainsi l'élément 0 du tableau correspond au clipboard des localisations, l'élément 1 au clipboard des sensations, l'élément 2 au clipboard des modalités, l'élément 3 au clipboard des concomitants.

```
typedef struct {
      char                    InMemory;
      unsigned short          NbSpt;
      char                    *Name;
      SYMPTOM_IN_MEMORY       *First;
      SYMPTOM_IN_MEMORY       *Last;
} SYMPTOM_CLIPBOARD;
SYMPTOM_CLIPBOARD SC[]
```

Cette structure est celle définie dans RADAR. Précisons que forcément il y a des champs qui nous intéressent moins pour notre application. Néanmoins, elle est assez riche pour être adoptée.

Nous en arrivons à la collecte proprement dite. Pour rappel, il s'agit pour le médecin homéopathe, aidé dans sa tâche par des outils de recherche, de placer les symptômes souhaités dans le "clipboard" correspondant. Cette collecte se fait par l'appel de la fonction:

```
EffectiveTake                ( REPERTORY_WINDOW *, short )
```

```
{
        AlreadyInClipboard          ( int );
        LoadRemedInMemory           ( HRS_SYMPTOM *);
        UpdateSymptomClipboard      ( unsigned short, unsigned short);
}
```

La fonction `AlreadyInClipboard()` vérifie si le symptôme n'est pas déjà présent dans le "clipboard". Si oui elle envoie un message et la collecte n'aura pas lieu. Sinon on peut seulement charger en mémoire les remèdes associés à ce symptôme par la fonction `LoadRemedInMemory()` et mettre à jour par la fonction `UpdateSymptomClipboard()`, notre liste de symptômes qui voit un nouveau symptôme venir s'ajouter à sa suite.

Nous avons automatisé cet algorithme afin que le "drag" d'un symptôme vers un "clipboard" se fasse dans le bon clipboard. Ainsi on testera en premier le nom du répertoire pour savoir si on est dans le répertoire de Boenninghausen. Si oui, on s'assurera qu'il s'agit des clipboards des localisations, des sensations ou des modalités. Auquel cas le numéro de clipboard est égal au numéro de chapitre du symptôme. Notons que ce dernier test ne concerne pas le clipboard quatre ou le clipoboard des concomitants, car le chapitre concomitants n'existe pas dans le répertoire. Et d'après Boenninghausen ces symptômes concomitants sont soit des symptômes coexistants, soit des symptômes ayant une relation de temps ou de circonstance, soit des symptômes d'aggravations ou d'améliorations, soit des modifications d'humeur. En somme le clipboard des concomitants peut recevoir des localisations, des sensations ou des modalités.

```
if  (strncmp(rep->RepAbbrev, "BG2",strlen("BG2"))==0) {

            if( (DefaultClipboard ==1) ||
                (DefaultClipboard ==2) ||
                (DefaultClipboard ==3)  )

                _scnumber = RecSymptom.NrChapter;

            else _scnumber = DefaultClipboard;
}
else _scnumber = DefaultClipboard;
```

Après avoir gardé nos différents symptômes dans les clipboards correspondants, nous avons la possibilité de les visualiser. Choisissez **Boenninghausen symptoms** dans le menu **Affichage**. Cela se fait par l'appel de la fonction `InitializeViewBoenWindow ( REPERTORY_WINDOW *, int )` qui crée la fenêtre *Boenninghausen symptoms* et ses objets de contrôle.

6.5 LA FENETRE BOENNINGHAUSEN SYMPTOMS

La fenêtre *Boenninghausen symptoms* se présente ainsi:

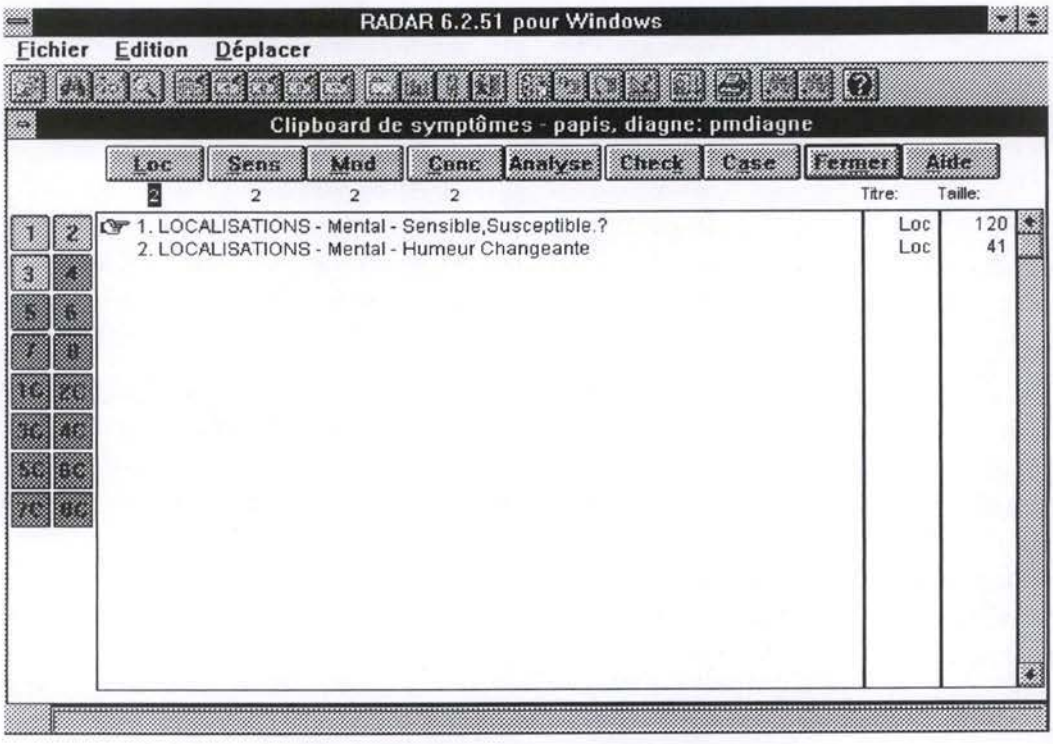


Figure 6-8 Fenêtre Boenninghausen symptom

Pour visualiser les symptômes de notre répertorisation, il suffit de cliquer sur l'un des boutons suivants:

- ◊ Loc affiche les symptômes dans localisation (ou dans le clipboard 1).
- ◊ Sens affiche les symptômes dans sensation (ou dans le clipboard 2).
- ◊ Mod affiche les symptômes dans modalité (ou dans le clipboard 3).
- ◊ Conc affiche les symptômes dans concomitants (ou dans le clipboard 4).

La fonction `ChangeViewBoenClipboard ( WINDOW, int )` permet de changer de vue, de passer d'un clipboard à un autre.

```
ChangeViewBoenClipboard ( WINDOW, int )
 {
        ...;
        DisplayViewBoenWindow ( WINDOW, int );
 }
```

La fonction `DisplayViewBoenWindow()` garnit notre fenêtre. Elle dessine le rectangle des symptômes, le rectangle des chapitres, et le rectangle des remèdes. Cette fonction affiche le texte des symptômes, le libellé du chapitre, et le nombre de remèdes dans ce symptôme.

6.5.1 LA VALORISATION DES SYMPTOMES

Le médecin, après avoir créé une répertorisation, devra valoriser ses symptômes. Il s'agit d'associer le même numéro à tous les symptômes entrant en compte dans le même symptôme complet. Cependant, un symptôme peut apparaître dans un ou plusieurs symptômes complets. Cette dernière considération nous laisse penser qu'une variable simple ne serait pas adéquate, mais qu'une variable composée (un tableau d'entiers par exemple) sera plus conforme. Dans la pratique, on se limitera au plus à huit symptômes complets pour un cas clinique.

Pour associer un numéro à un symptôme:

◊ Choisissez **Donner un numéro** dans le menu **Edition**, une fenêtre *Number symptom* s'affiche.

◊ Taper un numéro entre 1 et 8.

◊ Choisissez **OK** ou **Annuler** pour confirmer ou infirmer votre choix.

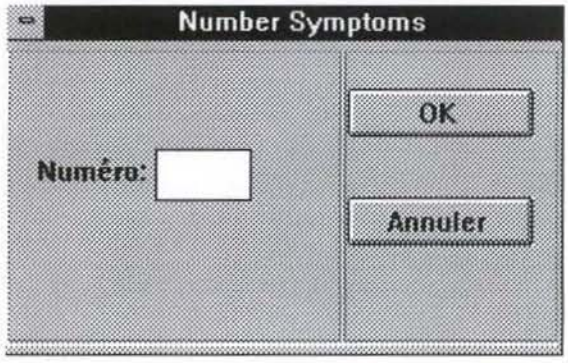


Figure 6-9 Fenêtre Number symptom

La même fenêtre sert aussi à saisir un chiffre pour supprimer un symptôme de notre symptôme complet. Pour effacer un numéro choisir **Effacer numéro** dans le menu **Edition**.

Dans notre fenêtre *Boenninghausen symptom* on l'aura remarqué, nous trouvons une liste de boutons. Chaque bouton est caractérisé par deux états:

◊ 1 état **OFF** voulant dire que ce numéro n'entre pas en compte dans notre symptôme.

◊ 1 état **ON** signifiant que ce numéro est attribué à notre symptôme.

Pour un symptôme courant pointé par le "finger", on sait voir tous les numéros qui lui sont associés. Il suffit de voir tous les boutons qui sont à **ON**.

La structure d'un bouton est définie delà sorte:

```
/*---- DEFINE SECTION ----------------------*/
#define    ON                              1
#define    OFF                             0

/*---- TYPEDEF SECTION ---------------------*/
```

```
typedef struct {
   int                Button;
   RCT                Rect;
} RECT_BUTTON;
RECT_BUTTON RectButton[];
```

où `RectButton[].Button` est une variable pouvant être à **ON** ou à **OFF** (plus exactement à 0 ou à 1), et `RectButton[].Rect` est un rectangle qui définit la position du bouton.

Un bouton est matérialisé par deux fichiers image bmp correspondant aux états **ON** et **OFF**. Par exemple pour le bouton 1, nous avons les deux fichiers suivants:

◊ cl1_0.bmp :  pour l'état **OFF**

◊ cl1_1.bmp :  pour l'état **ON**.

En associant un numéro `_number` à un symptôme nous mettons la variable `RectButton[_number].Button= ON` .La fonction `DisplayVBoenButton()` charge le bon fichier image et l'affiche à une position définie par `RectButton[_number].Rect` .

```
void DisplayVBoenButton ( _number )
int      _number;
{
    unsigned char  _name [ 255 ] ;

    sprintf (_name, "cl%d_%d.bmp", _number, RectButton[_number].Button );
    .... ;
}
```

L'important dans cette fonction est de voir la manière dont on charge le bon fichier image. L'instruction `sprintf` va charger dans `_name` le nom du fichier défini par `cl%d_%d.bmp` en remplaçant dans cette expression les `%d` par la valeur des variables `_number`,et `RectButton[_number].Button` dans l'ordre cité. Anisi par exemple si les variables `_number`,et `RectButton[_number].bouton` valent 1et 0, le fichier image `cl1_0.bmp`  sera chargé et affiché.

Une autre manière d'associer un numéro à un symptôme ou de supprimer un numéro rattaché à un symptôme, c'est de cliquer sur un bouton. Ceci a pour effet de mettre le bouton à **ON** s'il était à **OFF**, ou bien à **OFF** s'il était à **ON**.

Les boutons de  à  sont réservés aux clipboards des localisations, des sensations, et des modalités.

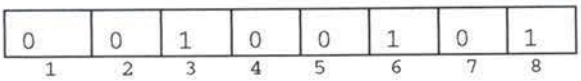
Les boutons de  à  sont réservés au symptômes concomitants.

Nous faisons cette différenciation pour mettre en valeur les symptômes de la maladie et les symptômes du malade.

Les explications fournies jusqu'à présent sont d'un niveau purement graphique, et épousent seulement le volet interface. Nous expliquons maintenant l'algorithme permettant d'associer ou de supprimer un numéro.

Nous avons implémenté le numéro de symptôme complet de façon à pouvoir profiter des possibilités offertes par le langage C en matière d'opérateurs logiques. En particulier, le C permet d'effectuer, sur des entiers, des caractères des opérations logiques bit à bit (ET, OU, NON, OUX) et les décalages gauche et droit d'un nombre donné de bits(ce qui revient à la multiplication ou à la division entière par des puissances de 2). Un type caractère est codé sur huit bits, et suffit pour gérer les numéros associer à un symptôme. Nous l'expliquons dans l'exemple suivant:

Exemple:



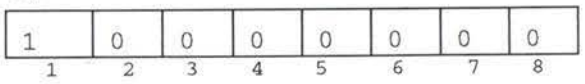
Les huit positions correspondent aux huit bits nécessaires pour coder un caractère. L'idée serait de mettre le bit à 1, pour une position donnée qui n'est rien d'autre que le numéro de symptôme complet. Nous lirons sur ce tableau que les numéros 3, 6, 8 sont associés à notre symptôme.

Dans la structure de donnée décrite plus haut, les champs `Degrees`, `Qualif`, serviront à donner des numéros à nos symptômes. Nous décrivons ci-dessous les algorithmes permettant d'associer, et d'éliminer un numéro de symptôme.

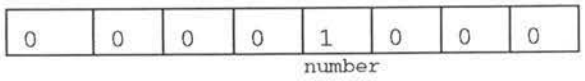
```
_c2=1;
_c2 = _c2 << number;
_spt->Qualif |= _c2;
```

Au début notre variable `_c2` est initialisée à 1.Ce qui revient à mettre à 1 le bit de la première position, et 0 partout ailleurs. L'instruction `_c2 = _c2 << number` va décaler le 1 de la première position de `number` position. La dernière instruction fait un OU bit à bit de la variable `_c2` avec notre numéro de symptôme `_spt->Qualif` .

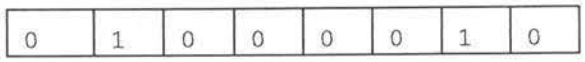
`_c2=1;`



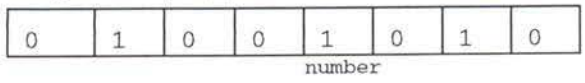
`_c2 = _c2 << number;`



`_spt->Qualif (par exemple)`



`_spt->Qualif |= _c2`

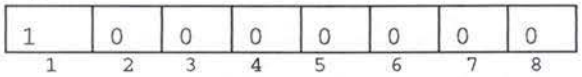


Si l'on permet d'associer un numéro à un symptôme, on devrait pouvoir aussi retirer un symptôme d'un symptôme complet. En d'autres termes, ce serait pour un symptôme, effacer un numéro attribué.

```
_c2=1;
_c2 = _c2 << number;
if(_spt->Qualif & _c2) {
      _spt->Qualif = _spt->Qualif & ~_c2;
}
else {
      ...
}
```

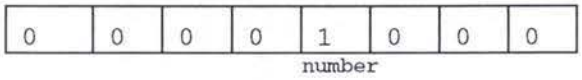
On initialise `_c2` à 1

`_c2=1;`



On décale la valeur 1 de la première position de `number` position

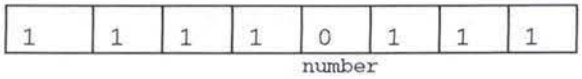
`_c2 = _c2 << number;`



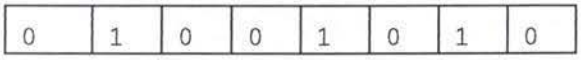
Le test `if(_spt->Qualif & _c2)` vérifie si le numéro à effacer est un numéro attribué.

Pour effacer le numéro, on fait le complément à 1 de la variable `_c2` , et un ET avec notre numéro de symptôme.

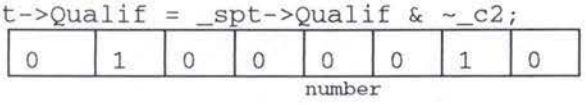
`~_c2`



`_spt->Qualif (par exemple)`



`_spt->Qualif = _spt->Qualif & ~_c2;`



Remarque:

- On peut associer à plusieurs symptômes le même numéro. La démarche reste le même, sauf qu'il faut selectionner plusieurs plusieurs symptômes. Pour selectionner plusieurs symptômes appuyez sur la touche shift et utilsez la touche fléche vers le bas.
- Pour effacer tous les numéros associer à un symptôme, choisir **Effacer tous** dans le menu **Editer**.

6.5.2 LA FONCTION CHECK COMPLETE SYMPTOM

Comme nous l'avons illustré dans un chapitre précedent (chapitre Première approche), la fonction check complete symptom a pour rôle essentiel de guider l'utilisateur. En effet au risque de se noyer dans le fratas de symptômes complets, cette fonction

permettra au médecin homéopathe de voir parmi ces symptômes complets ceux qui ne sont pas complets.

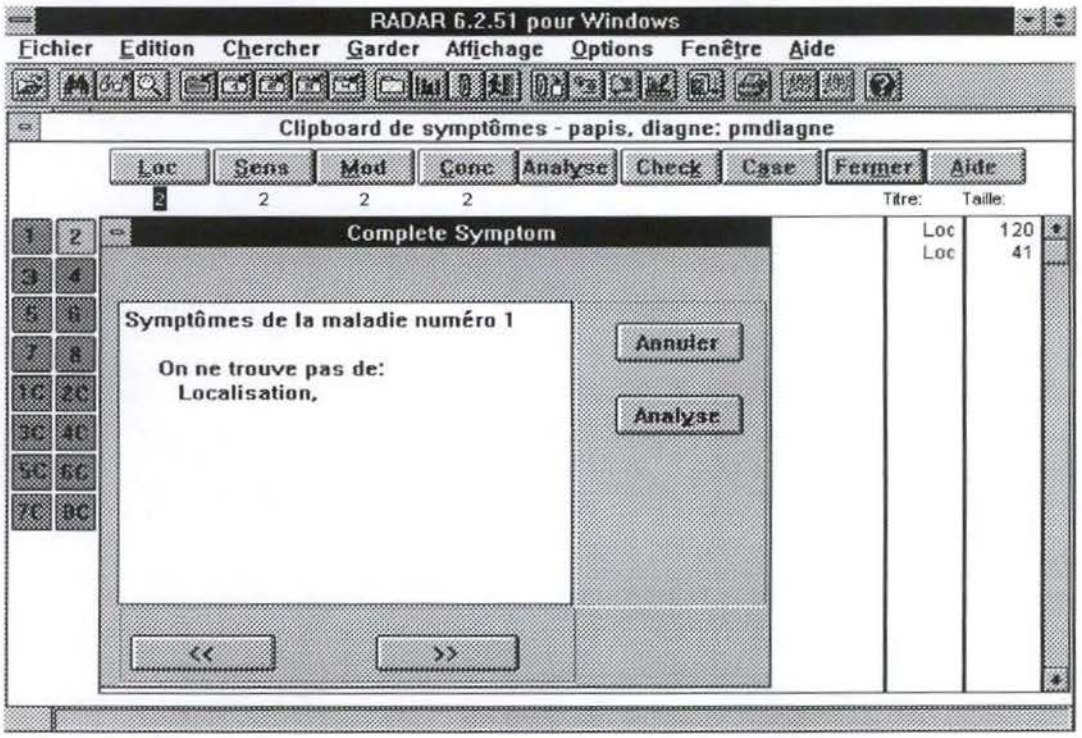


Figure 6-10 Fenêtre Check complete symptom

Il s'agit pour chaque numéro de symptôme complet de voir si l'on trouve au moins un symptôme dans chaque catégorie à savoir une localisation, une sensation, et une modalité. Comme l'illustre la figure, la fenêtre *Check complete symptom* envoie un message pour signaler les caractéristiques manquantes.

Pour lancer le Check à partir de la fenêtre *Boenninghausen symptom* , il suffit de cliquer sur le bouton Check.

La fonction `InitializeViewCheckWindow ( )` crée la *Check complete symptom*, et ses objets de contôle. Nous analysons à présent la structure de donnée associée à notre fonction Check:

```
/*-------define section ----------*/

#define   LOCALISATION 1
#define   SENSATION    2
#define   MODALITE     4
#define    CONCOMITTANT 8

/*-------type section --------*/

typedef struct {

            unsigned char Present;
```

```
        unsigned char Complet;
        unsigned char Check;
        } COMPLET_SYMPTOM;
COMPLET_SYMPTOM  CC[];
```

- Le champ Present permet de dire qu'un numéro est présent ou pas. Si le numéro a été attribué une fois à un symptôme sa valeur est égale à 1, 0 sinon.
- Le champ `Complet` permet de dire qu'un numéro est complet ou pas. Il vaut 1 dans le cas complet, 0 sinon.
- Le champ `Check` permet de voir, pour un numéro donné, les caractéristiques qui sont présentes ou absentes. Ce champ est marqué d'une valeur 1 ou 0 en des positions bien précises:

Loc		Sens		Mod			

Ainsi pour voir si notre numéro est complet, il suffit de voir si `Loc` est à 1, `Sens` est à 1, `Mod` est à 1. Si un des trois est à 0, nous pouvons déterminer ce qui fait défaut dans notre symptôme complet.

Nous présentons l'algorithme qui calcule la valeur des champs cités ci-dessus:

```
void  CheckCompleteSymptom()
{
```

Pour un symptôme donné:

```
c1 = _Spt->Qualif;
```

On regarde tous les numéros qui lui sont associés:

0	1	0	0	0	0	1	0

```
for ( num = 1; c1 != 0; c1 >>= 1) {
```

On teste les cases qui sont à 1

```
if ( c1 & 1) {
```

Si une des cases est à 1 le numéro est présent

```
CC[num-1].Present = 1;
```

La fonction `CheckType()` retourne le type de clipboard (une localisation, une sensation, modalité). La case correspondant au type de clipboard dans notre champ `Check` est positionnée à 1.

```
CC[num-1].Check  |= CheckType(_SCNumber);
```

Notre numéro est complet si toutes les cases correspondant aux différents types de clipboards dans notre champ `Check` sont positionnées à 1.

```
                if( ((CC[num-1].Check) & (LOCALISATION)) &&
                  ((CC[num-1].Check) & (SENSATION)) &&
                  ((CC[num-1].Check) & (MODALITE)) )
                  CC[num-1].Complet = 1;
                else  CC[num-1].Complet = 0;
                }
        num++;

        }
```

Après avoir rempli les champs de la structure de données associées au module Check. La fonction DisplayCheckWindow() se chargera d'afficher les messages signalisant au médecin les caractéristiques manquantes dans ses symptômes complets.

```
DisplayCheckWindow(_number)
int _number;
{
```

Nous calculons la valeur des champs de notre structure de données pour un numéro de symptôme complet donné _number :

```
CheckCompleteSymptom();
```

Nous testons si le numéro en question est présent et non complet :

```
If ((CC[_number].Present == 1) && (CC[_number].Complet == 0)) {
```

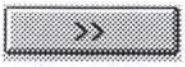
Nous testons par la suite les caractéristiques qui manquent. La fonction DisplayTextEdit() affiche le message correspondant.

```
        If ( ((CC[_number].Check) & (LOCALISATION)) == 0) {
        DisplayTextEdit() ;
        }

        if ( ((CC[_number].Check) & (SENSATION)) == 0 ) {
              DisplayTextEdit() ;
        }

        if ( ((CC[_number].Check) & (MODALITE)) == 0  ) {
              DisplayTextEdit() ;
        }
}
```

Sur notre fenêtre *Check complete symptom* les boutons :

◊ ◊ 

(Previous) nous sert à revenir en arrière dans notre recherche de symptômes incomplets.

◊ ◊  (Next) nous permet d'avancer dans notre recherche de symptômes incomplets.

Ainsi nous avons une vue en défilement ponctuel sur tous les numéros de symptômes incomplets.

Le médecin, étant seul juge de sa répertorisation et de la valorisation de ses symptômes, pourra s'il le souhaite, revenir soit à la collecte de symptômes, ajouter un autre symptôme, soit tout simplement valoriser un symptôme déjà présent mais qui n'a pas été associé à son symptôme complet.

6.5.3 EDITER UN SYMPTOME COMPLET

Une autre manière sans doute plus convaincante de se fixer sur l'état incomplet d'un symptôme, serait d'afficher tous les symptômes entrant en compte dans ce symptôme complet.

Pour afficher un symptôme complet, nous faisons un **Ctrl + Clique** sur un bouton désignant un numéro de symptôme complet.

La fonction `DisplayCompleteSymptom(_number)` affiche les symptômes associés à un numéro de symptômes complets.

```
void  DisplayCompleteSymptom(_number)
int _number;
{
```

Le clipboard 5 nous sert de tampon, l'idée est de copier dans ce clipboard tous les symptômes qui portent le numéro `_number` .

Nous prenons la précaution de vider le clipboard tampon avant d'y copier des symptômes. La fonction `FreeSptInMemory()` efface tous les symptômes du clipboard 5.

```
FreeSptInMemory(5,0);
```

Pour les clipboards localisation, sensation, modalité, et concomitant ; nous testons les symptômes qui portent le numéro `_number`.

```
for(_SCNumber=0; _SCNumber<4; _SCNumber++) {
            if (_number >0 && _number <9) {
                c1 = 1;
```

Dans notre variable `c1` nous mettons 1 à la position définie par `_number`.

```
c1 = c1 << (_number);
```

Nous testons si `_number` est attribué à notre symptôme.

```
if( _Spt->Qualif & c1) {
```

Nous copions le symptôme dans notre clipboard tampon.

```
CopySptToSC(5,_SCNumber,_Spt);
```

```
                    }
        }
}
```

Dans la fonction `CopySptToSC()` nous copions aussi le numéro de clipboard. Ceci nous permet de connaître la caractéristique du symptôme si nous avons une localisation, une sensation, une modalité, ou concomitant. Dans notre *Boenninghausen symptoms* nous avons une colonne titre qui indique la caractéristique du symptôme.

Pour afficher notre symptôme complet, nous appelons la fonction `ChangeViewBoenClipboard()`.

6.5.4 EDITER LE CAS

La fonction éditer un symptôme complet vu précedemment est un cas particulier de notre cas présent. Lorsque nous éditons le cas, nous éditons tous les symptômes complets. Nous appelons la fonction `DisplayCompleteSymptom(_number)` dans une boucle.

```
For (_number=1; _number<9; _number++){
DisplayCompleteSymptom(_number);
}
```

La fonction `ChangeViewBoenClipboard()`se chargera d'afficher les symptômes complets.

Pour éditer le cas complet il suffit de cliquer sur le bouton Case dans notre fenêtre *Boenninghausen symptom.*

6.6 ANALYSE

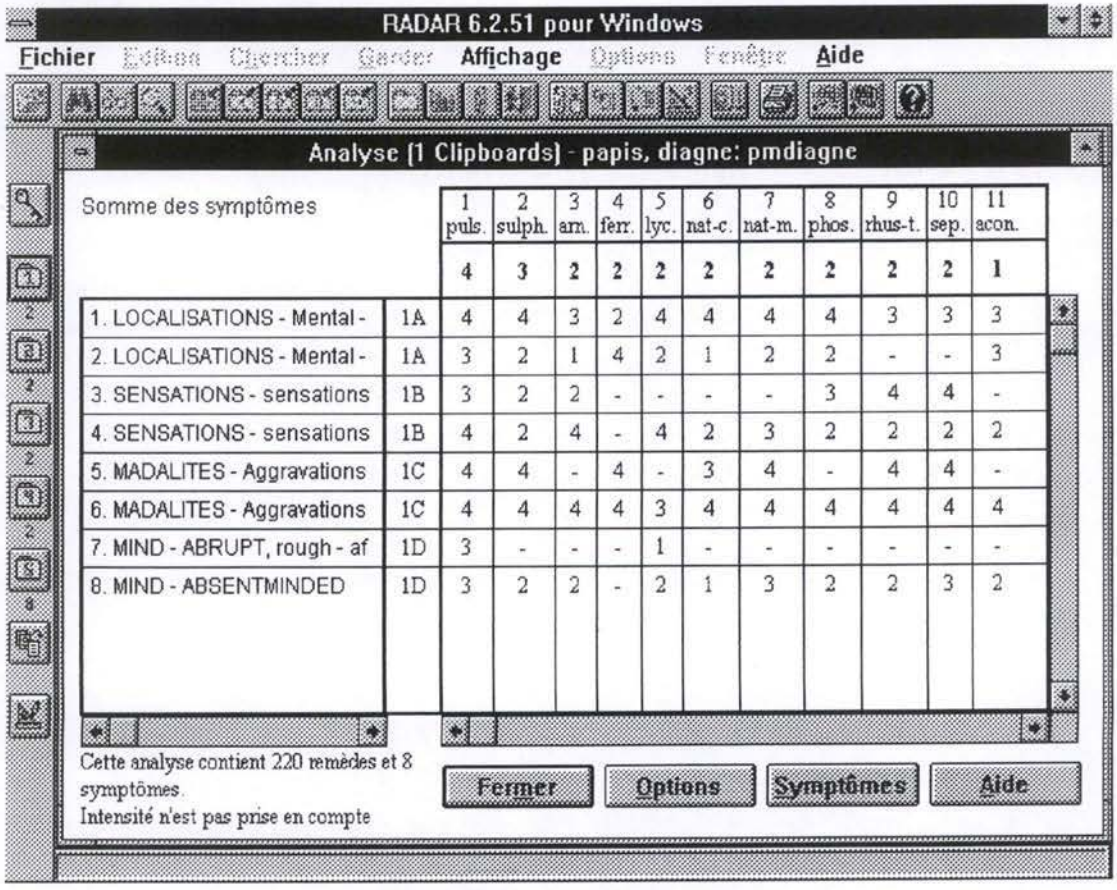


Figure 6-11 Fenêtre VboenAnalyse

L'analyse ne différe pas de l'analyse qui se faisait dans RADAR. Dans RADAR nous avons le concept de groupe qui permet de rassembler des symptômes afin de créer une nouvelle rubrique. Dans cette nouvelle rubrique, les remèdes sont toujours comptés selon le plus haut degré de leur occurrence parmi les différents symptômes. Une analyse standard serait la somme des symptômes qui affiche le nombre de fois que le remède apparaît dans les différents symptômes. Dans notre cas présent, l'échelle à considérer n'est plus le symptôme, mais plutôt le groupe.

Cette idée correspond bien à la méthode de Boenninghausen. Ainsi nous constituerons quatre groupes: le groupe des localisations, des sensations, des modalités, et des concomitants.

Dans RADAR les groupes sont désignés par des lettres de l'alphabet. Les localisations porteront la lettre A, les sensations la lettre B, les modalités la lettre C, et les concomitants la lettre D.

```
void NewVboenAnalyse()
{
```

```
int _SCNumber;
SYMPTOM_IN_MEMORY *_Spt;
```

Nous utilisons le clipboard 5 comme clipboard tampon. Neanmoins par précaution nous vidons de tout son contenu avant d'y copier nos différents groupes.

```
FreeSptInMemory(5,0);
for(_SCNumber=0; _SCNumber<4; _SCNumber++) {
```

Nous testons pour si le symptôme est valorisé autrement, il ne sera pas considéré dans notre analyse

```
if ( (_Spt->Qualif !=0)
    || (_Spt->Degrees !=0)){
```

Notons la ligne d'instruction qui suit, dans la structure `SYMPTOM_IN_MEMORY` associée à un symptôme nous avons un champ `Group` qui permet le regroupement. La valeur de ce champ est une lettre de l'alphabet. Pour rappel l'indice du clipboard des localisations est égal à 1, celui des sensations est ègal 2, celui des modalités est égal à 2, celui des concomitants est égal à 4. Ainsi dans cette affectation qui montre une fois de plus les possibités du langage C, `'A'+(_SCNumber-1)` vaut `A` si `_SCNumber= 1`, et `'A'+(_SCNumber-1)` vaut `B`, si `_SCNumber= 2`.

```
_Spt->Group = 'A'+(_SCNumber-1);
```

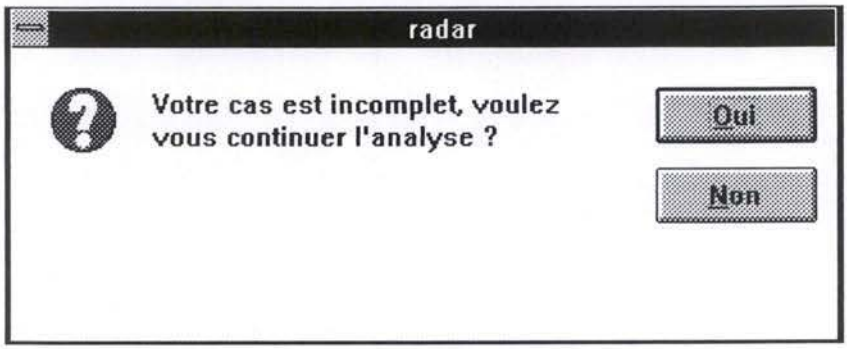
Nous copions le symptôme en question dans notre clipboard tampon qui nous servira pour notre analyse

```
CopySptToSC_5(_Spt);
}
}
}
```

RADAR nous permet de choisir le clipboard à analyser. Dans l'algorithme défini au-dessus, nos symptômes regroupés selon les caractéristiques (localisation, sensation, modalité, concomitant) sont enregistrés dans notre clipboard tampon. Nous imposerons ainsi au moment de l'analyse notre clipboard tampon.

Pour lancer l'analyse à partirde notre fenêtre *Boenninghausen symptom*, il suffit de cliquer sur le bouton Analyse.

Nous prenons soin de vérifier, et de signaler au medecin homéopathe l'état de son cas. Si ses symptômes sont complets, l'analyse se déroulera sans intervention du système. Autrement nous signalons par la fenêtre ci-dessous, l'état du cas. Nous devons apporter une aide au diagnostic, le médecin, étant seul juge, pourra s'il le souhaite continuer son analyse; S'il désire suspendre l'analyse, la fenêtre *Check complete symptom* s'affichera; et il pourra vérifier son cas.



6.7 EDITER LE CAS COMPLET

Cette derniére partie sert de résumé à notre application. RADAR gère un fichier patient, et une fiche de consultation comme l'illustrent les deux figures ci-dessous.

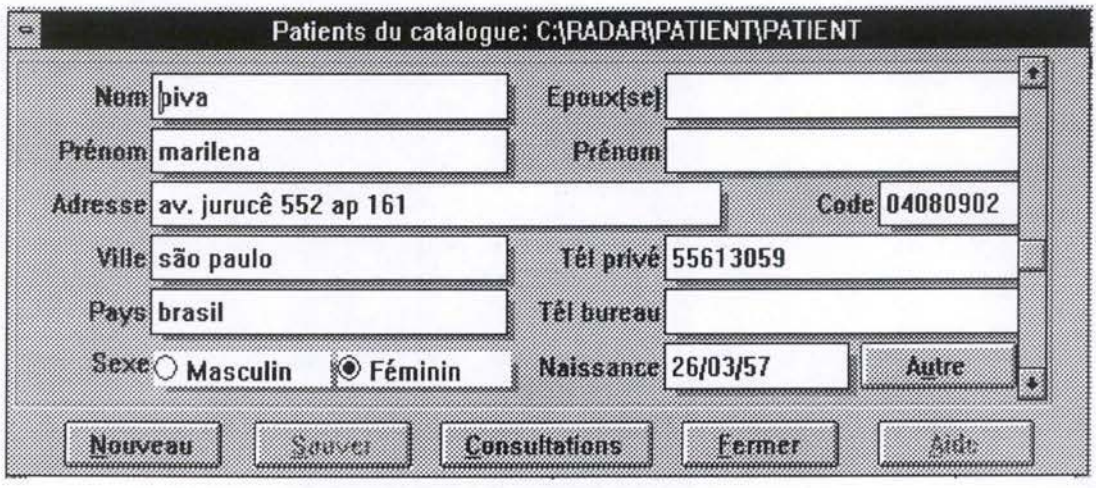


Figure 6-12 Fenêtre Patient

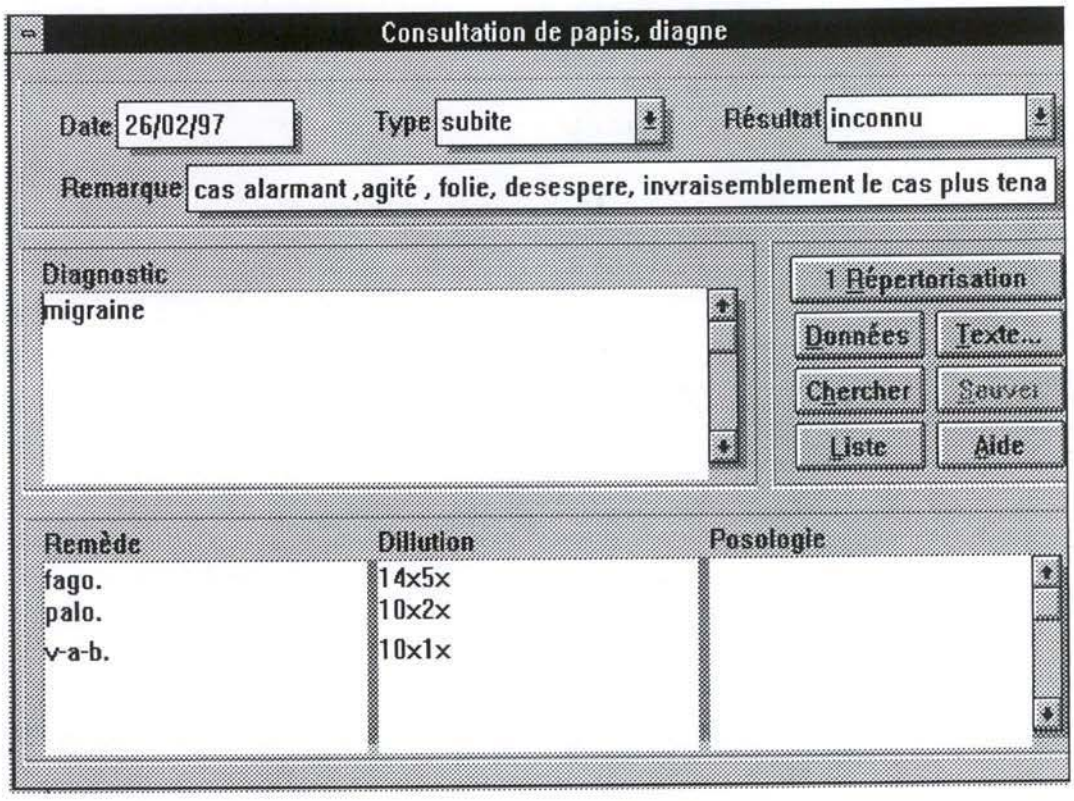


Figure 6-13 Fenêtre Consultation

La fenêtre *Clinical Case* est intéressante pour le medecin homéopathe. Elle donne une vue globale du cas. Elle reprend les données du fichier patient, et du fiche de consultation.

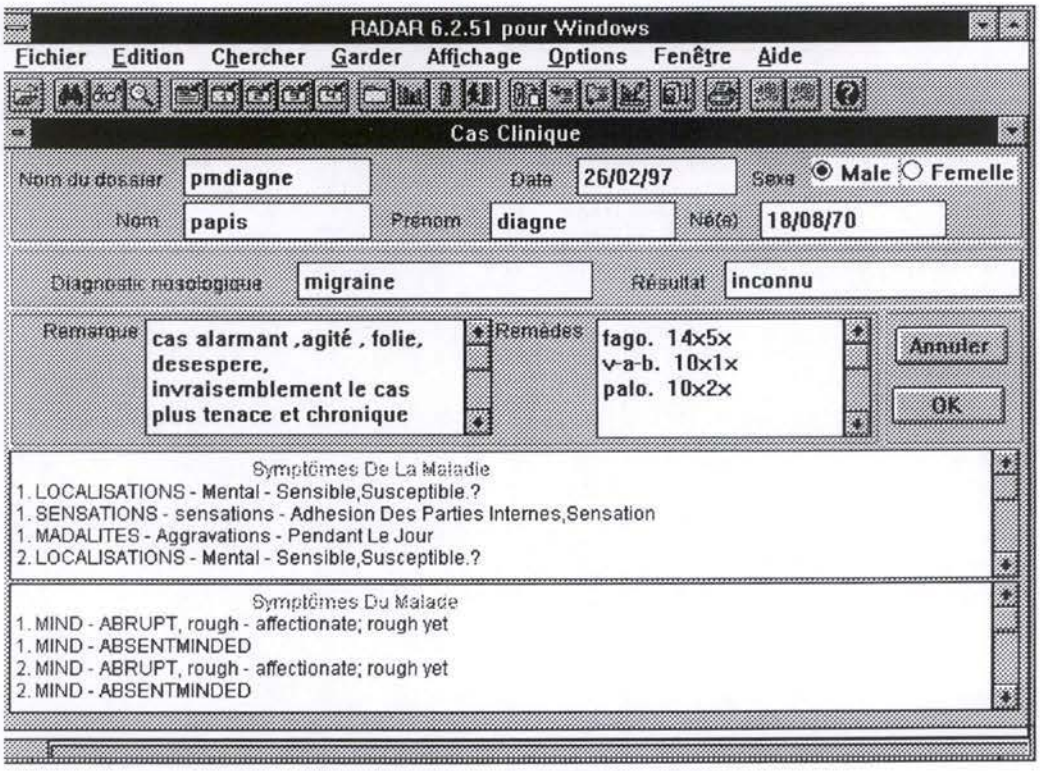


Figure 6-14 Clinical Case

Etudes de cas

7. ETUDE DE CAS

7.1 CAS 1

Ce cas est tiré d'un article de A. Jacques intitulé : Intoduction à l'homéopathie Hahnemanienne.

7.1.1 HISTOIRE DU PATIENT

Nombreux épisodes d'amygdalite ayant nécessité l'amygdalectomie dans l'enfance.

Ulcère bulbaire (1964) et infection urinaire la même année

Depuis août 1974, fréquentes infections respiratoires.

Le 20 février 1975, le patient présenté des urines hématuriques (urines brun foncé) pendant deux jours, immédiatement après un état de grippal avec une fièvre et pharyngite. Une observation en milieu internistique a permis de retenir le diagnostic de glémorulo-néphrite aiguë hémorragique compliquant une pharyngite aiguë.

7.1.2 AFFECTION ACTUELLE

Au terme de l'hospitalisation, il fut décidé que le malade suivrait un traitement antibiotique visant à prévenir le rechutes; en particulier on lui injecta tous les quinze jours, par voies intramusculaires, PENADUR L.A soit 1 200 000 UI de benzathine pénicilline G, dont l'action retard prolongée est particulièrement bien adaptée aux infections exigeant des concentrations sanguines durables.

Le 15 mai 75, un examen clinique approfondi avait démonté la persistance d'une discrète hématurie, de traces de protéinurie, d'un taux élevé d'antistreptolysines.

Malheureusement il récidivera le 10. 07. 75 malgré le traitement chronique imposé en février.

7.1.3 EXAMEN CLINIQUE (10. 07. 75) CLASSIQUE

7.1.3.1 FROTTIS DE GORGE

- La culture démontre la pullulation de streptocoques hémolytiques.
- L'antibiogramme permet de constater une résistance au PENSTAPHO ainsi qu'au LINCOCIN.

7.1.3.2 EXAMEN DES URINES

- Albumine : 0,80 gr/litre
- Globules blancs : 88/champs
- Globules rouges : 100/champs.
- Culture stérile.

7.1.3.3 EXAMEN SOMATIQUE

- Hyperthermie : 39,9° C
- Adénopathies cervicales bilatérales
- Gorge pourpre

7.1.4 EXAMEN CLINIQUE HOMEOPATHIQUE

7.1.4.1 LOCALISATIONS

- Extrême sensibilité du cou au toucher :
  - ◊ `EXT. THROAT SENSITIVE to slightest touch.`

Pour chercher un symptôme, nous commençons par voir le chapitre où il est inscrit, nous sélectionnons ce chapitre. Sur la *Figure 7-1* nous voyons toute une série d'images matérialisant une zone bien particulière du corps humain. Ces images constituent les différends chapitres du répertoire. Dans notre cas présent, le chapitre à considérer est `EXTERNAL THROAT`. Sur notre fenêtre, un chapitre sélectionné est inscrit dans un carré.

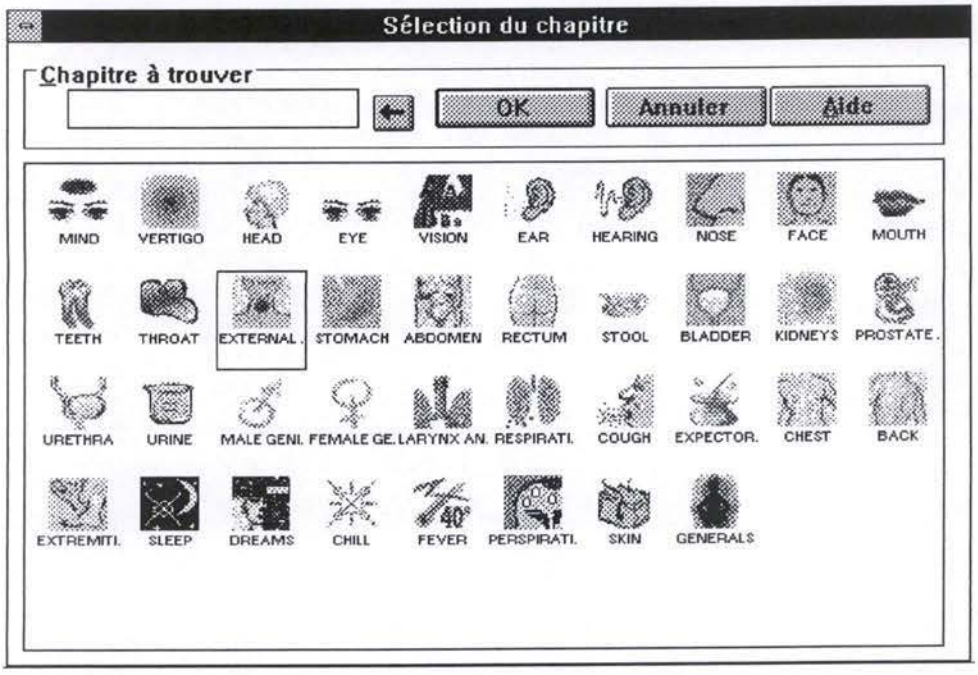


Figure 7-1 Fenêtre des chapitres

Nous débutons la recherche du symptôme. La fenêtre *Recherche symptôme* s'affiche, les lettres de l'alphabet nous servent d'index. On notera qu'il n'est pas nécessaire de former entièrement le nom du symptôme, dans notre fenêtre en cliquant sur la seule lettre s qui débute le nom de notre symptôme, nous voyons déjà apparaître le symptôme cherché `EXT. THROAT SENSITIVE`.

Une fois le symptôme trouvé, nous pouvons l'enregistrer dans le clipboard correspondant, dans notre cas présent nous enregistrons le symptôme `EXT. THROAT SENSITIVE` to slightest touch dans le clipboard des localisations.

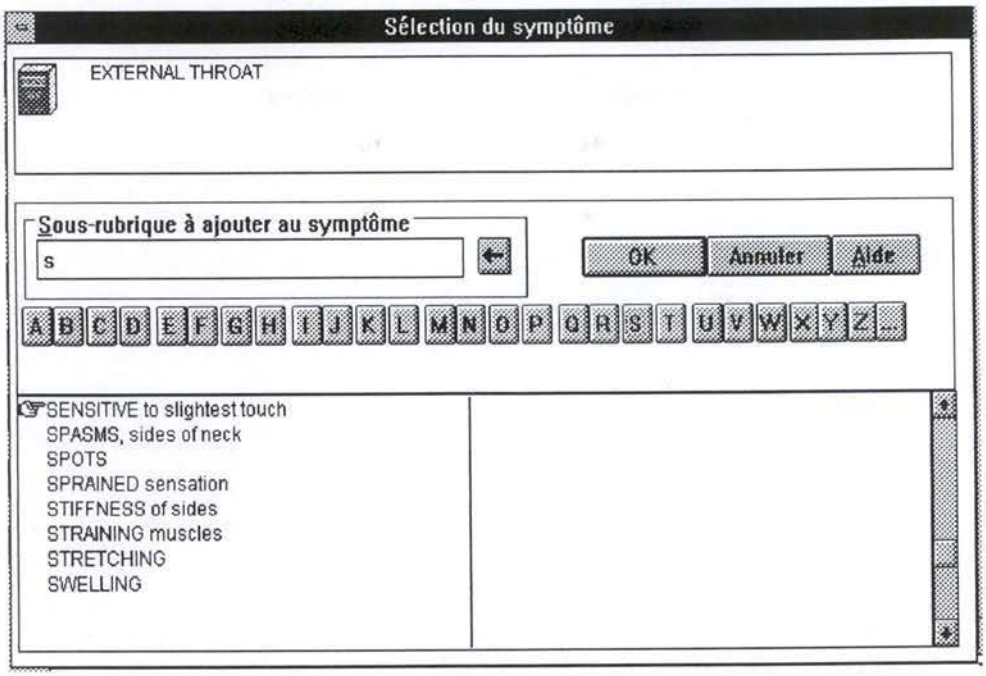


Figure 7-2 Recherche du symptôme EXT. THROAT SENSITIVE to slightest touch.

Dans les étapes suivantes, nous procéderons de la même manière pour trouver un symptôme. Les figures *Figure 7-1* et *Figure 7-2* nous montrent comment il faut procéder. Nous nous passerons d'afficher à chaque fois ces deux fenêtres.

- Adénopathies cervicales :
  - ◊ `EXT. THROAT SWELLING cervical glands.`
- Difficulté pour ouvrir la bouche :
  - ◊ `MOUTH open, difficult to.`
- Haleine épouvantable :
  - ◊ `MOUTH odor, offensive.`
- Difficulté pour tirer la langue :
  - ◊ `MOUTH protruded tongue difficulty, with.`
- Le malade essaie de caler sa langue entre les dents :
  - ◊ `MOUTH protruded difficulty, catches on the teeth.`
- La langue tremble surtout quand il la tire, et est animée de mouvements rapides d'avant en arrière :
  - ◊ `MOUTH trembling tongue when protruding it`
  - ◊ `MOUTH protuded rapidly, darting in and out, like a snake's`
- Aphte sur le bout de la langue, ainsi que dans la bouche :
  - ◊ `APHTAE tongue tip`
- Le fond de la gorge est pourpre :
  - ◊ `THROAT discoloration purple`
- Crevasses entre les orteils :
  - ◊ `EXTREMITIES cracked skin toes, between`
- Excoriations entre les orteils :
  - ◊ `EXTREMITIES excoritions toes, between`

Une fois la collecte de cette première partie finie, nous pouvons aller voir dans notre *View Boenning* les symptômes enregistrés dans localisation. Dans la *Figure 7-3* nous lisons 12 symptômes, le chiffre tout juste en dessous du bouton **LOC** nous donne le nombre de symptômes présent dans le clipboard. Plus loin nous lisons la taille de chaque symptôme qui correspond au nombre de remèdes présents dans le symptôme. En double cliquant avec la souris sur la taille du symptôme par exemple sur le chiffre 4 associé au symptôme `EXT. THROAT SENSITIVE` , une fenêtre contenant la liste des remèdes s'affiche, comme l'illustre notre *figure 6-4*. Sur cette fenêtre, notre symptôme `EXT. THROAT SENSITIVE` a une taille de 4 et contient un remède de degré 1 (`Lach.`), 2 remèdes de degré 2 (`Lac-c.`, `Nicc.`), et un remède de degré 3 (`Bell.`).

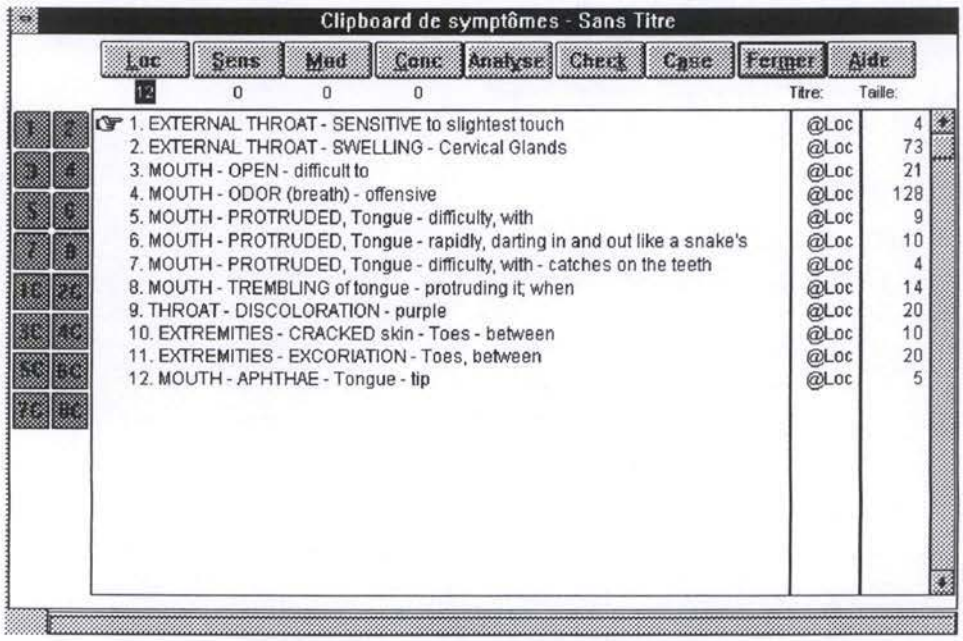


Figure 7-3 Répertorisation des localisations

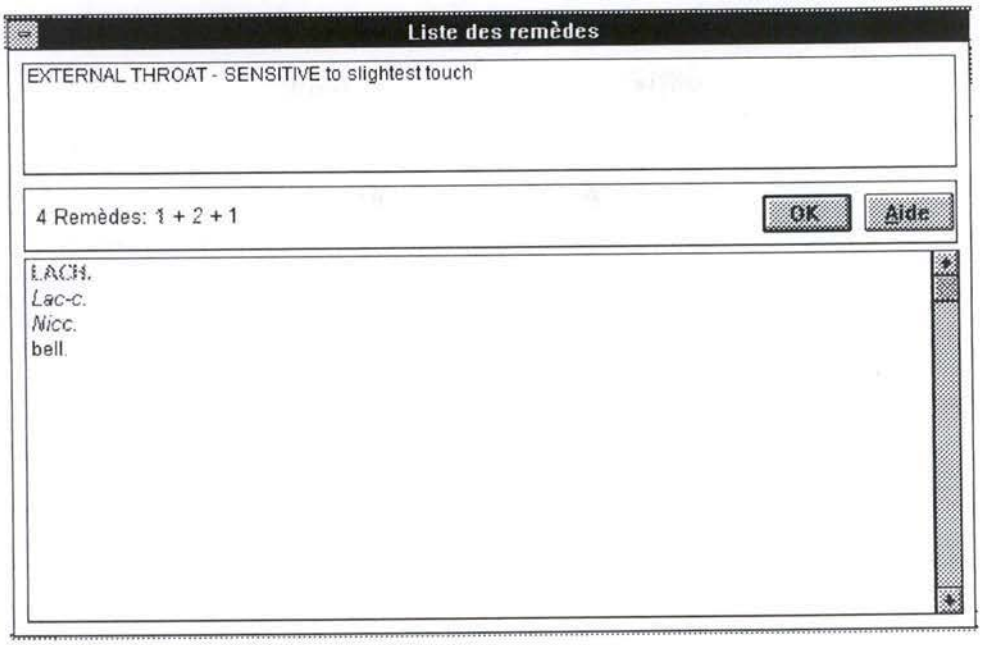


Figure 7-4 Liste des remèdes

7.1.4.2 SENSATIONS

- La douleur ressentie dans la gorge est décrite comme s'il s'agissait d'une plaie à vif située à gauche :
  - ◊ `THROAT pain sore left`
- L'interessé a peine à avaler :
  - ◊ `THROAT swallowing, difficult`
- D'abord, sa salive, ainsi que :
  - ◊ `THROAT swallowing, difficult saliva`
- Les liquides bien plus que les aliments solides :
  - ◊ `THROAT swallowing, difficult liquids more difficult than solids`
  - ◊ `THROAT pain swallowing, liquids`
- En particulier, déglutir des boissons chaudes est intolérables :
  - ◊ `THROAT pain warm drinks`
- Les boissons froides font du bien :
  - ◊ `THROAT pain cold drinks amel`
- L'intéressé souffre de devoir continuellement se racler la gorge, car il éprouve la sensation de présence de miettes de pain dans celle-ci :
  - ◊ `THROAT bread crumbs, sensation of`
  - ◊ `THROAT pain hawking, on`
- Le malade est extraordinairement loquace depuis que la fièvre s'est installée et a pris une allure typhique :
  - ◊ `MIND loquacity heat, during`
- Cela est d'autant plus paradoxal qu'il éprouve d'énormes difficultés à s'exprimer, car prétend-t-il, sa langue est très lourde; de plus la gorge est très sèche :
  - ◊ `MOUTH speech difficult typhoid, in`
  - ◊ `MOUTH speech difficult heaviness of tongue`
  - ◊ `THROAT dryness`

- Il n'avalera que de très petites quantités de liquide froid redoutant l'acte de déglutition :
  - ◊ `STOMACH thirst heat during`
  - ◊ `STOMACH thirst small quantities`
  - ◊ `STOMACH thirst dread liquids, with`

Nous affichons à présent, les symptômes dans le clipboard des sensations. Nous comptons 16 symptômes pour le cas traité (voir *figure 6-5).*

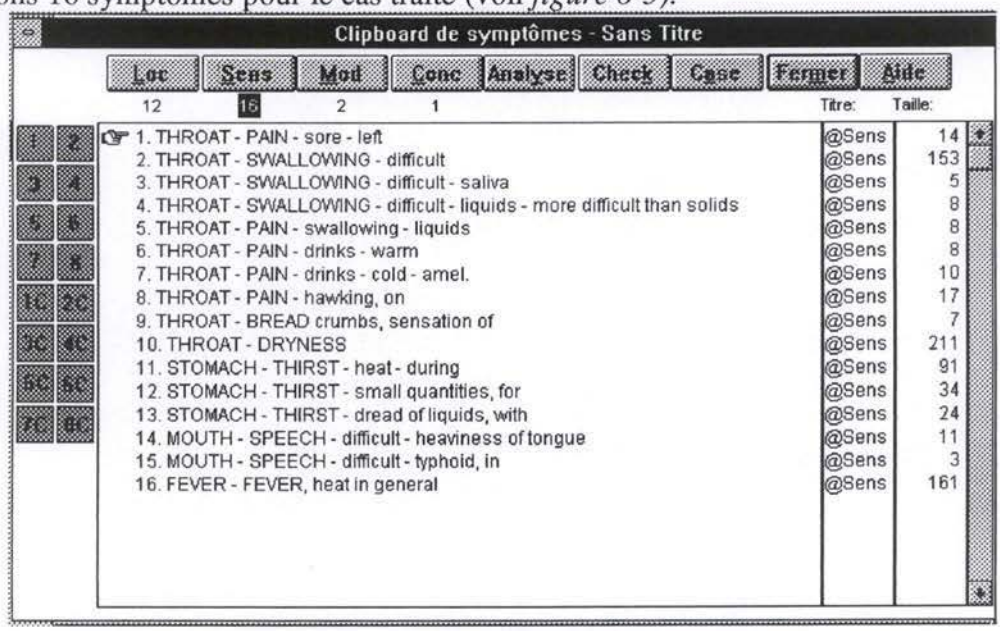


Figure 7-5 Répertorisation des sensations

7.1.4.3 MODALITES

- L'intéressé ne supporte pas le port d'un col serré, ni d'une cravate ce qui est à l'origine d'ennuis professionnels, vu qu'il doit porter l'uniforme. Il a l'impression d'étouffer :
  - ◊ `EXTERNAL THROAT clothing, agg`
  - ◊ `THROAT choking clothing agg`

Nous affichons à présent, les symptômes dans le clipboard des modalités. Nous comptons 2 symptômes pour le cas traité (voir *figure 6-6).*

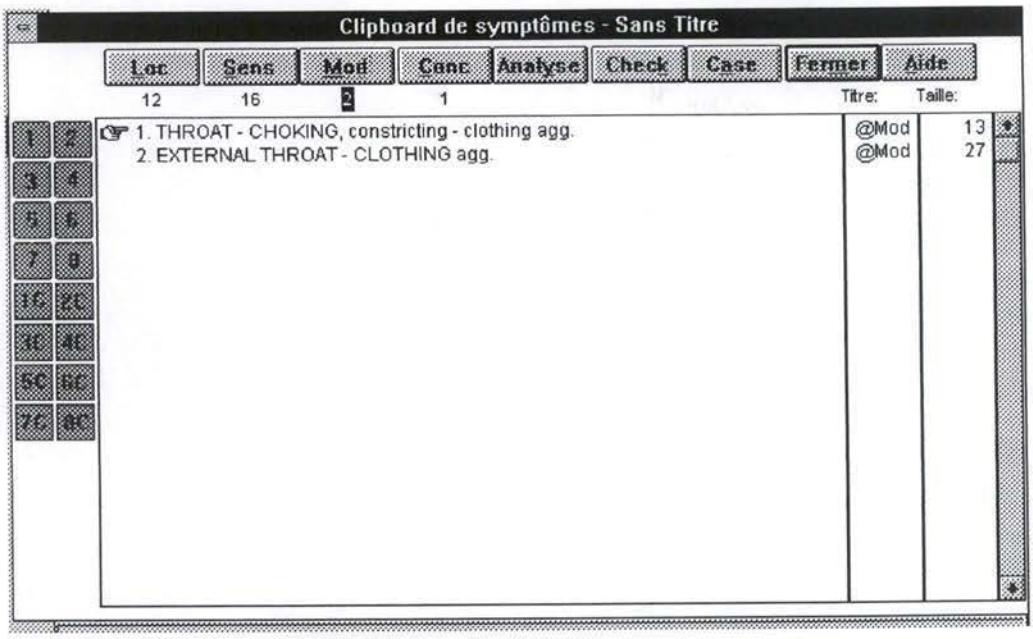


Figure 7-6 Répertorisation de modalités

7.1.4.4 SYMPTOMES CONCOMITANTS

- Des proches le décrivent comme un personnage jaloux. En particulier très attiré par les femmes maigres; il interdirait à la sienne de maigrir, de peur probablement, dans son esprit, qu'elle devient l'objet de convoitise des autres. Par ailleurs, l'expérience a montré qu'il a systématiquement pris un jour de congé pour accompagner son épouse à la consultation du médecin :
  - ◊ `MIND jeaslousy as foolish as it is irresistible`

Nous affichons à présent, les symptômes dans le clipboard des concomitants. Nous comptons 1 symptôme pour le cas traité (voir *figure 6-7).*

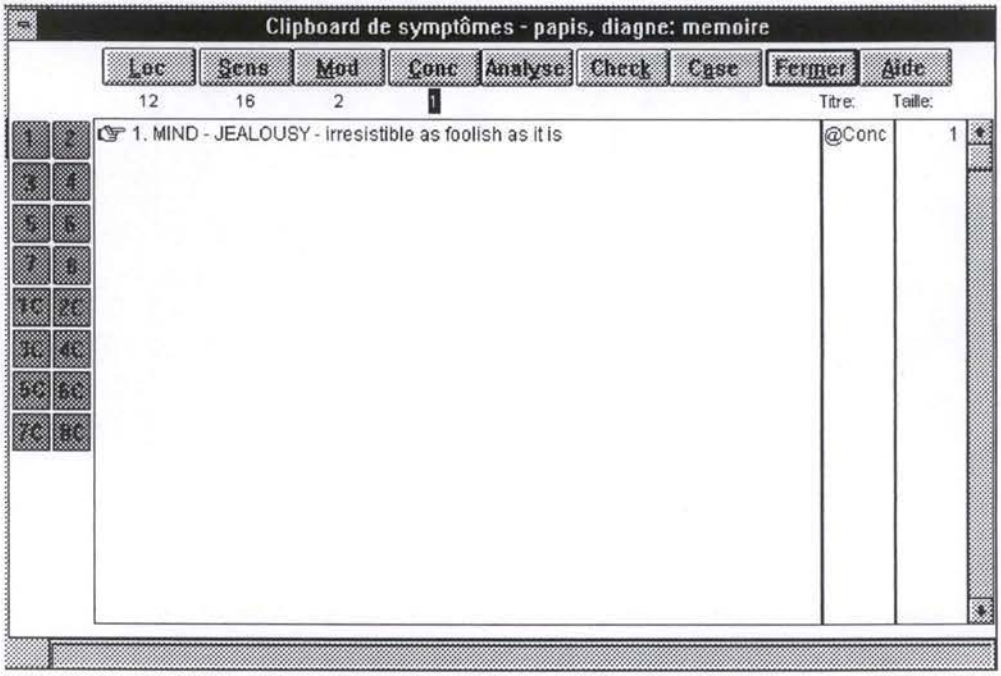


Figure 7-7 Répertorisation des concomitants

7.1.4.5 VALORISATION DE NOTRE CAS

Nous allons à présent constituer le tableau symptomatique du patient. Nous sommes en face d'un cas clinique où l'ensemble des symptômes entre dans un seul symptôme complet. Ainsi pour valoriser nos symptômes nous leur associerons le numéro 1.

Nous sélectionnons l'ensemble des symptômes dans localisation, à cet ensemble nous donnons le numéro 1, en cliquant tout simplement sur le bouton 1. Sur la *figure 6-7* nous voyons que le bouton 1 est actif.

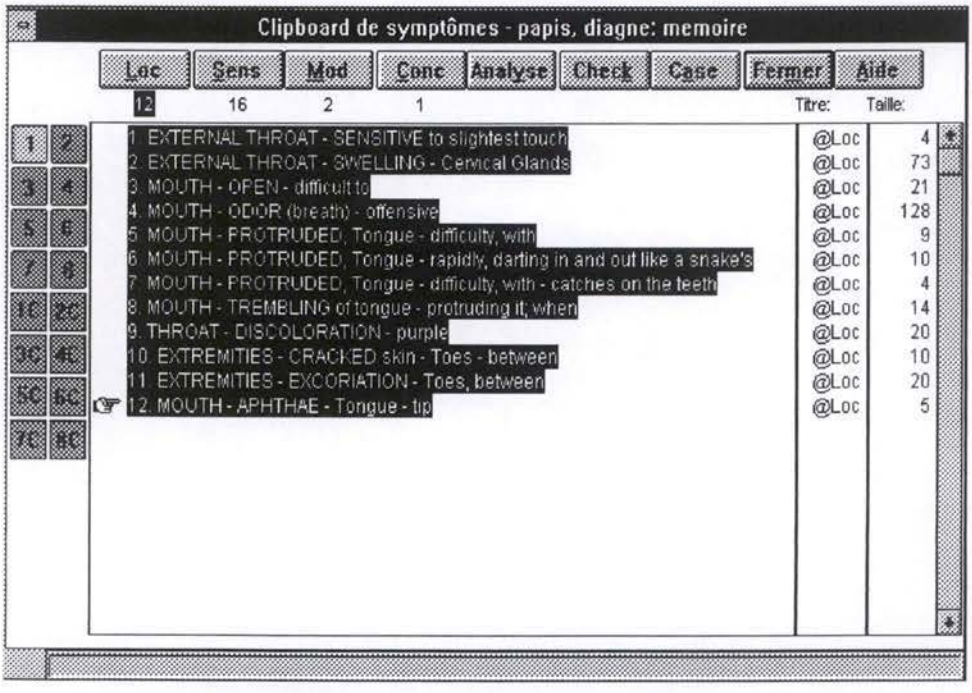


Figure 7-8 Valorisation des symptômes dans localisation

A cette étape, si nous essayons de voir l'état notre de symptôme complet nous aurons le résultat suivant comme l'indique la*figure 6-9*.

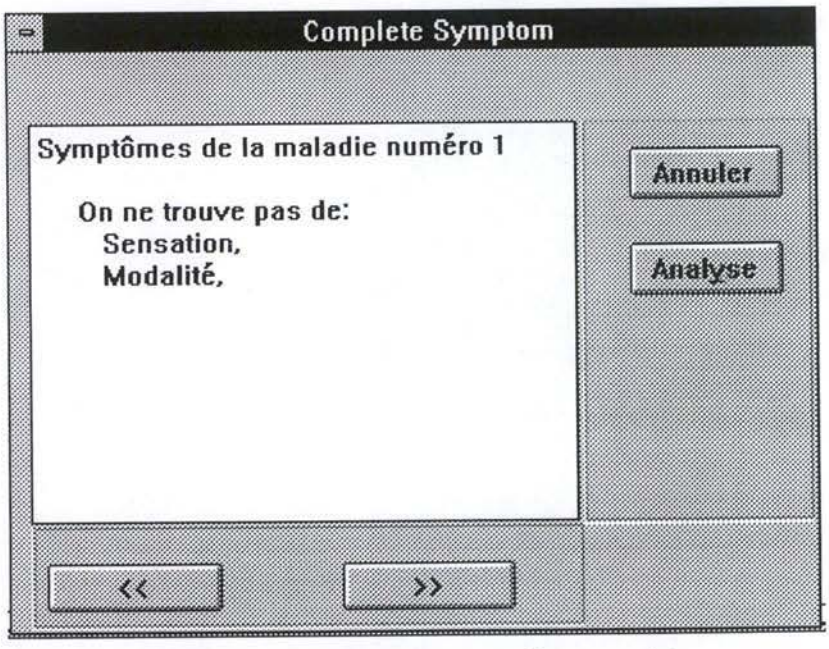


Figure 7-9 Vérification du symptôme complet

Le résultat indiqué par la *figure 6-9* ne nous surprend pas. En effet nous n'avons pas procédé à la valorisation des autres clipboards à savoir les modalités et les sensations.

Une remarque peut être faite à ce niveau notre fonction `CheckCompleteSymptom()` ne nous signale pas l'absence des symptômes concomitants. Ceci pour deux raisons, d'une part les symptômes concomitants ne font partie des symptômes de la maladie, et d'autre part ces symptômes ont pour rôle essentiel d'affiner le choix des remèdes.

Dans l'étape qui suit nous allons compléter notre cas en valorisant les symptômes présents dans les autres clipboards.

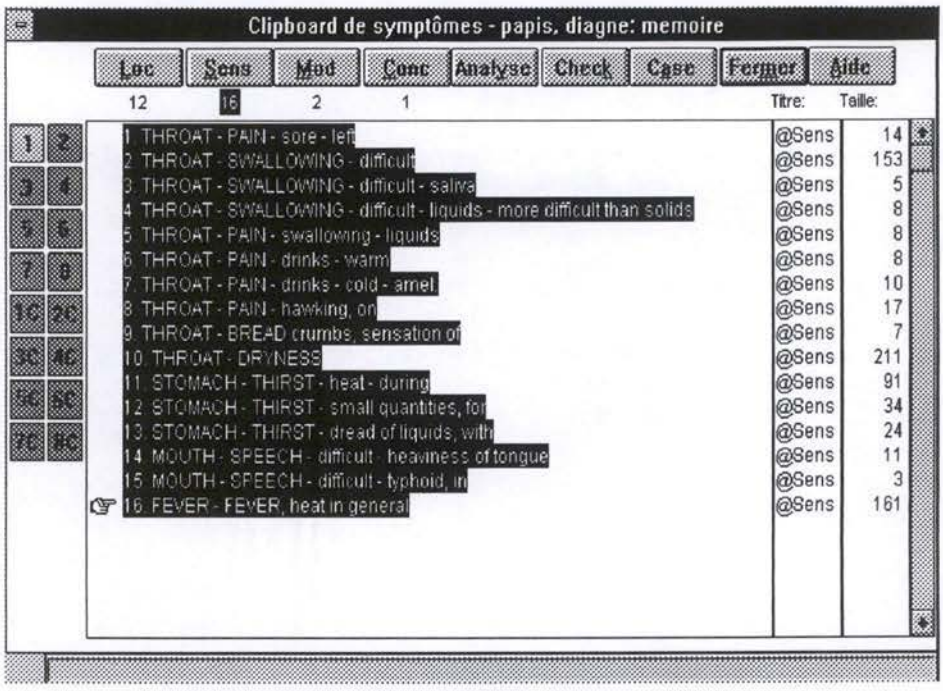


Figure 7-10 Valorisation des sensations

Nous procéderons de la même manière pour valoriser les symptômes dans modalités.

Notre cas est complet, les trois caractéristiques sont présentes. Pour s'en assurer nous pouvons faire appel au `Check`, la *figure 6-11* nous signale le résultat.

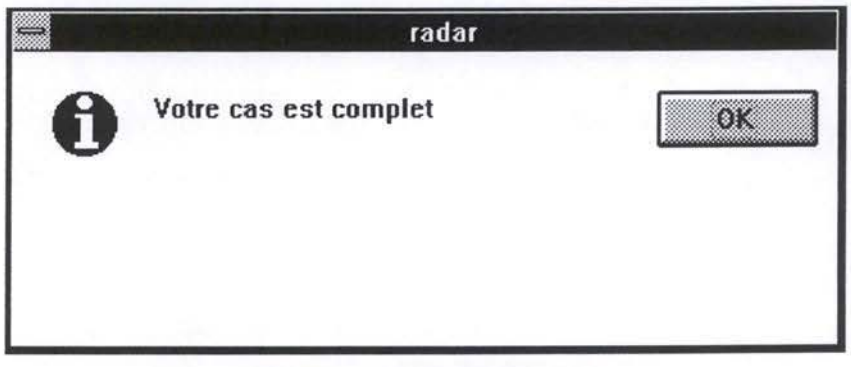


Figure 7-11 Vérification du cas

Pour les symptômes concomitants nous avons vu que les boutons marqués d'un chiffre et de la lettre C, leur sont réservés. Néanmoins pour rattacher un symptôme concomitant à un symptôme complet, le chiffre marqué sur le bouton nous sert de référence. Ainsi pour associer notre symptôme concomitant au cas présent, nous donnerons le numéro 1C; 1C voulant dire, le ou les symptômes concomitants associés au symptômes complet 1.

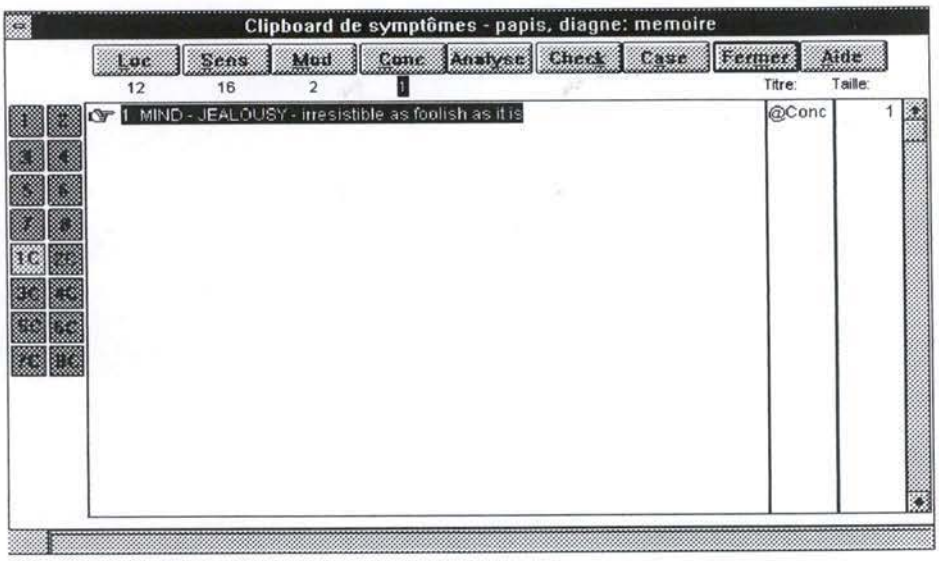


Figure 7-12 Valorisation des concomitants

7.1.4.6 ANALYSE DU CAS

L'examen de la *figure 6-13* montre que l'analyse contient 319 remèdes et 31 symptômes. Lachesis a un poids total de 11, ce qui est de loin le plus important comparé aux autres remèdes.

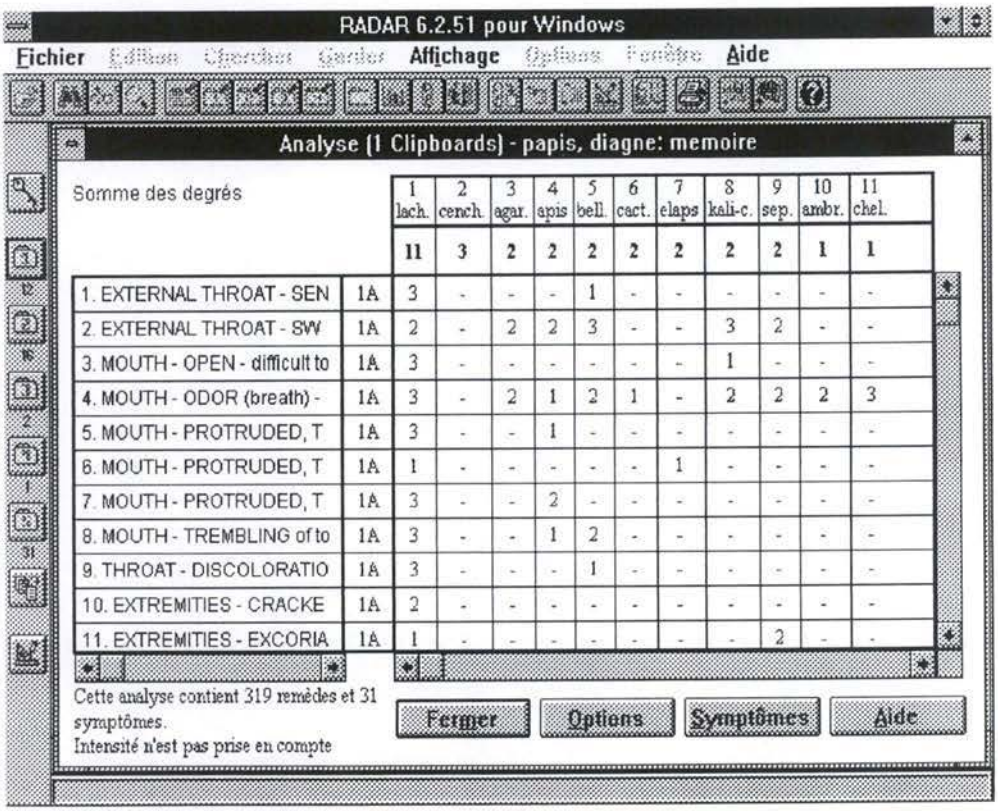


Figure 7-13 Analyse somme des degrés

Nous avons aussi poussé l'analyse pour connaître les remèdes qui couvrent les quatre caractéristiques à savoir les localisations, les sensations, les modalités, et les concomitants. L'analyse de la *figure 6-14* nous montre que Lachesis couvre ces 4 caractéristiques, et que les autres remèdes ne couvraient partiellement qu'une caractéristique particulière.

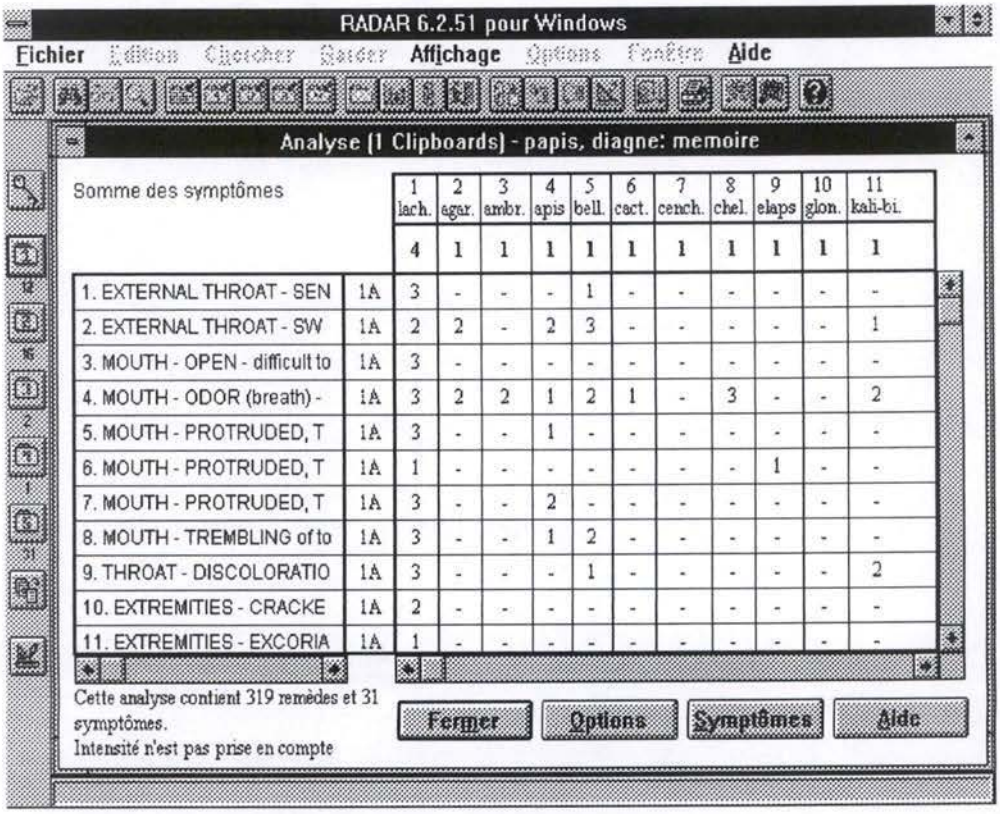


Figure 7-14 Analyse sommes des symptômes (méthode Boennighausen)

L'analyse combinée de ces deux méthodes nous amène à dire que Lachesis couvre la totalité des caractéristiques avec un poids de 11 degré. Dès lors Lachesis apparaît sans aucun doute comme le remède le plus indiqué.

Voyons maintenant ce que nous donnerait l'analyse avec Synthésis.

L'examen des *figure 6-15*, et *figure 6-16* montre que:

- ◊ Lachesis a un poids total de 76 et couvre la totalité des symptômes soit 31.
- ◊ Lycopodium couvre 16 symptômes avec un poids de 30
- ◊ Belladonna couvre 15 symptômes avec un poids de 32.
- ◊ Apis couvre 13 symptômes avec un poids de 22.

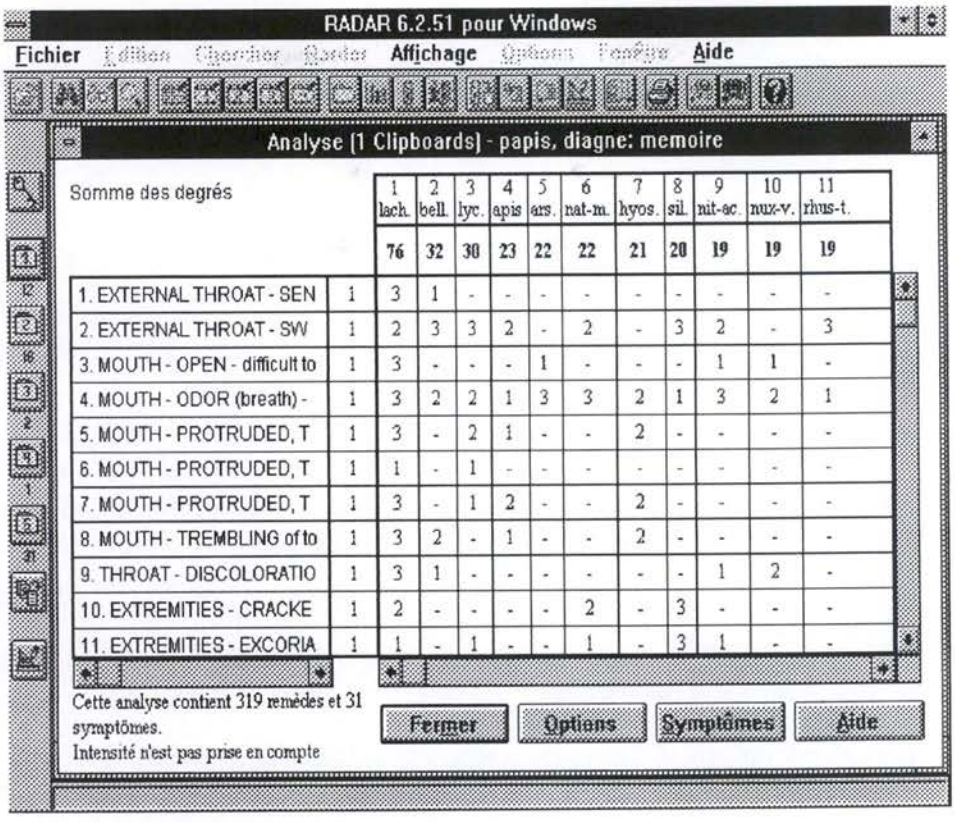


Figure 7-15 Analyse somme des degrés avec Synthésis

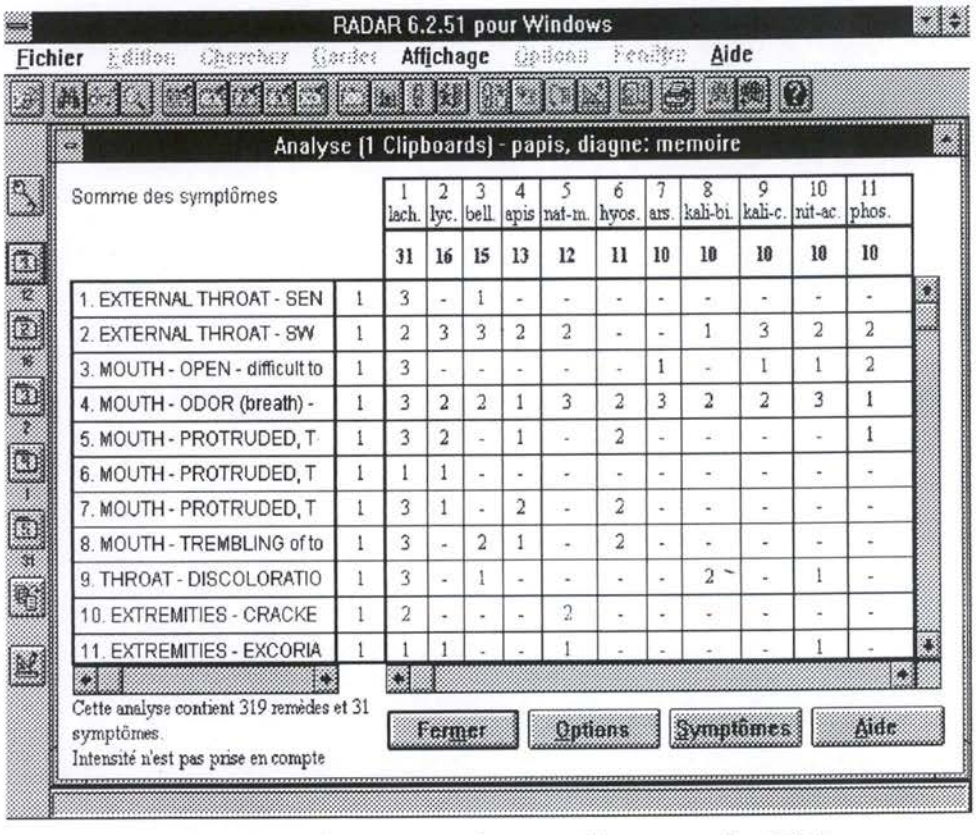


Figure 7-16 Analyse somme des symptômes avec Synthésis

Contrairement à l'analyse avec la méthode de Boenninghausen où le choix du remède est plus qu'évident. Ici, nous pouvons seulement affirmer que sur 319 remèdes, 4 seulement pourraient probablement guérir le malade à savoir Lachesis, Belladonna, Lycopodium, et Apis. Dès lors un diagnostic remédial s'impose.

7.1.4.7 DIAGNOSTIC REMEDIAL

Ayant procédé de la sorte, le médecin homéopathe pense avoir de bonnes raisons pour retenir ces quatre remèdes en question.

Afin d'éviter toute subjectivité et autres causes d'erreurs, il va opérer une double vérification de son hypothèse en consultant deux types ouvrages:

- Matière médicale pure: encyclopédie contenant tous les symptômes qu'ont présentés les expérimentateurs sains lors des provings de ces remèdes.
- Matière médicale clinique: encyclopédie contenant tous les symptômes qui ont permis à des praticiens de guérir différentes maladies avec ces remèdes.

Cette triangulation permet de découvrir le remède le plus semblable et dés lors de poser le diagnostic remédial.

7.1.4.8 PRESCRIPTION DU REMEDE

Dans le cas présent, Lachesis, remède apparemment le plus indiqué a été prescrit à la dose unique 10.000 K, ce qui signifie que le venin de ce terrible serpent a fait l'objet de 10.000 opérations successives de dilution-dynamisation préalablement à ce que le pharmacien délivre ce remède au malade.

Le malade fut débarrassé de son affection en quelques heures, et à l'heure actuelle il n'a pas encore récidivé.

7.2 ETUDE DE CAS 2

Ce cas a été présenté par le docteur G. Coquillart dans la revue Belge en 1994.

Il s'agit d'une dame âgée de 71 ans en consultation en mai 92.

7.2.1 ANTECEDENTS

- Adolescentes, nombreuses angines, et abscées amygdaliens.
- A 41 ans cure de varices.
- Fracture de la cheville droite.
- Double entorse.
- A 52 ans: hypertension soignée avec une association d'un diurétique d'épargne potassique et un thiazide.
- A 69 ans début de cervicalgies soignée avec de l'ibuprofène.

7.2.2 CONSULTATION HOMEOPATHIQUE

Elle consulte pour des douleurs aiguës dans la nuque et des céphalées occipitales.

C'est une personne très courageuse qui ne peut rester à ne rein faire; toujours à l'heure, même plutôt à l'avance.

Elle présente un tremblement dans les mâchoires et me dit éprouver des tiraillements et des élancements dans l'articulation temporo-maxillaire des deux côtés.

Elle a toujours faim, peut en trembler de même que d'émotion.

Elle aime les sucreries et surtout les légumes.

Toutes les plaintes actuelles ont débuté il y a deux ans suite à une agression qui a provoqué une grave peur.

Elle ne veut plus vivre seule et surtout ses maux s'aggravent lorsqu'elle est seule.

Elle qui était si bien ordonnée et calme, voit son caractère changer et son humeur devenir instable chaque fois qu'elle souffre.

7.2.3 REPERTORISATION FAITE SUR BOENNINGHAUSEN

7.2.3.1 SYMPTOMES CARACTERISTIQUES

- Localisation:
  - ◊ région occipitale externe "occiput".
  - ◊ Tremblement de la mâchoire inférieure "lower jaw".
- Sensation:
  - ◊ Douleur tiraillante et lancinante dans les articulations
- Modalité:
  - ◊ suite d'une très grande peur, frayeur "excitement, fight".
  - ◊ Aggravation dans la solitude "when alone".
- Concomitant:
  - ◊ instabilité, humeur variable "aternating mood"

7.2.3.2 EDITION DU CAS COMPLET

Après avoir valorisé nos symptômes, nous pouvons éditer le cas. Nous avons sauté cette étape, mais il s'agit de dresser le tableau symptomatique du patient. Ainsi on valorisera les symptômes présents dans les différents clipboards. Dans le cas présent, nous avons donné le numéro 1 aux symptômes présents dans localisation, sensation, et modalité, et le numéro 1C aux symptômes concomitants. La *figure 6-17* affiche le cas complet. Notre cas comporte deux localisations, une sensation, deux modalités, et un symptôme concomitant.

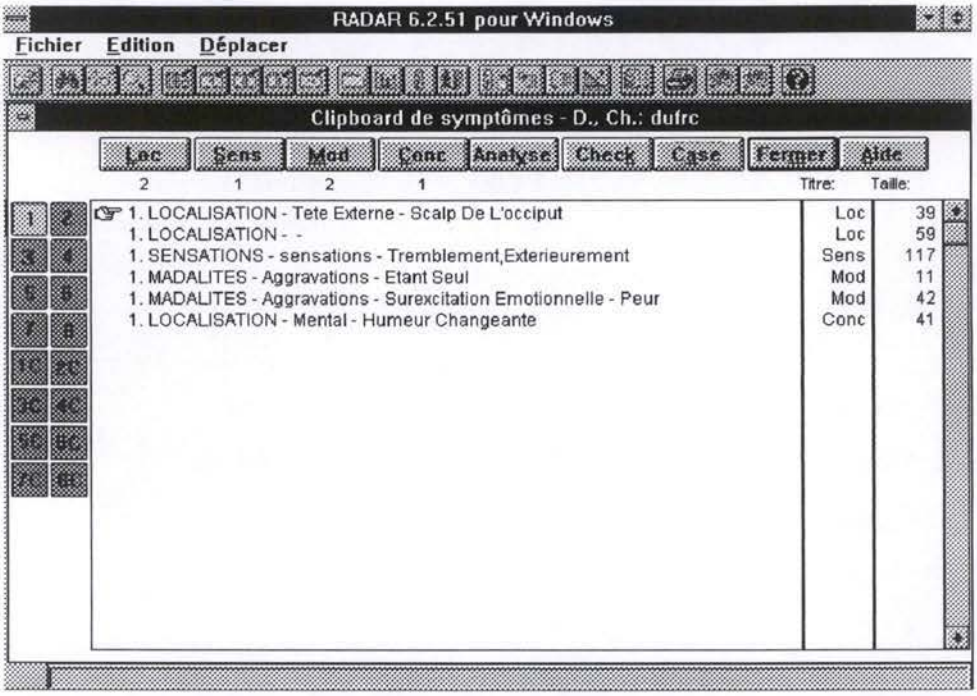


Figure 7-17 Edition du cas complet

7.2.3.3 ANALYSE DU CAS

Nous passons directement à la phase d'analyse. Notre cas étant considérablement petit, nous ne sommes pas obligés de passer par la phase de vérification. Il suffit de considérer notre *figure 6-17* qui affiche les symptômes du cas, pour s'apercevoir que toutes les caractéristiques requises pour l'état complet d'un symptôme, sont présentes.

L'examen de la *figure 6-18* montre que:

- ◊ Zinc a un poids total de 15 et couvre la totalité soit les 4 caractéristiques.
- ◊ Lycopodium couvre les 4 caractéristiques avec un poids de 14.

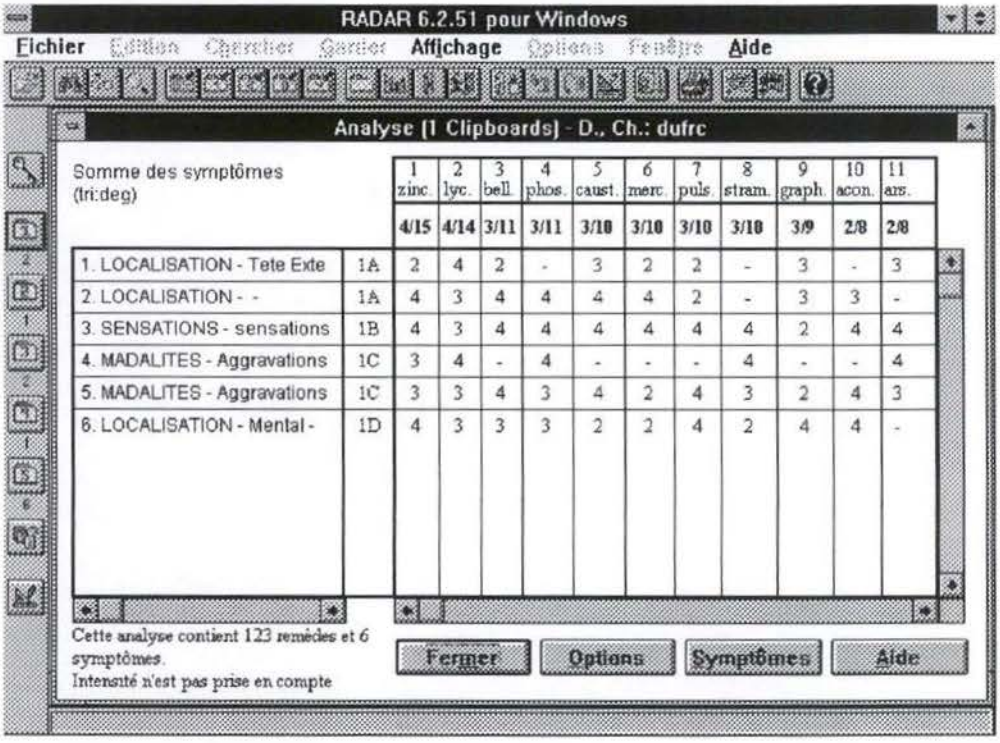


Figure 7-18 Analyse du cas

7.2.3.4 PRESCRIPTION DE REMEDES

- ◊ Zinc 200 a été retenu et prescrit au patient.
- ◊ Juillet 92 : les douleurs dans la nuque et la tête sont bien améliorées, mais les tremblements réapparaissent 1 semaine après la fin de la cure. Cet état est sans particularité pour l'âge et de plus l'examen clinique ne fait pas apparaître de Parkinson, pas de signe de la roue dentée. On lui prescrit du Zinc M.
- ◊ Septembre 92 : nette amélioration des tremblements de la tête et de la bouche, mais elle a fait une chute en rue et a une fracture de trois métacarpiens à la main droite. Début de cataracte à l'oeil droit. Dès qu'elle s'énerve ou doit attendre, les tremblements reviennent. On lui prescrit du Zinc 35.
- ◊ Fin novembre 92: nette amélioration des tremblements, il reste une gêne dans sa nuque; pas de tremblement de la tête, très léger à la lèvre inférieure. On lui prescrit de nouveau du Zinc 35.
- ◊ Mars 93 : périodes de plus en plus rares; retour des troubles de l'équilibre. On lui prescrit du Zinc M.
- ◊ Septembre 93 : disparition des troubles de l'équilibre, elle n'a plus eu de tremblement, rare gêne dans la nuque et disparition des céphalées. On lui prescrit cette fois-ci du Zinc 35.
- ◊ Mars 94 : Elle peut vivre seule sans panique et ses troubles sont peu gênants

7.2.3.5 EDITER LE CAS CLINIQUE

Cette dernière étape est intéressante pour le médecin homéopathe. En effet, la *figure 6-19* illustrant le cas clinique, reprend dans une seule fenêtre toutes les informations

au sujet du patient, le diagnostic établi par le médecin à une date de consultation précise, le remède retenu. Nous trouvons également dans cette fenêtre, les symptômes présents dans le cas, à savoir les symptômes de la maladie, et les symptômes du malade.

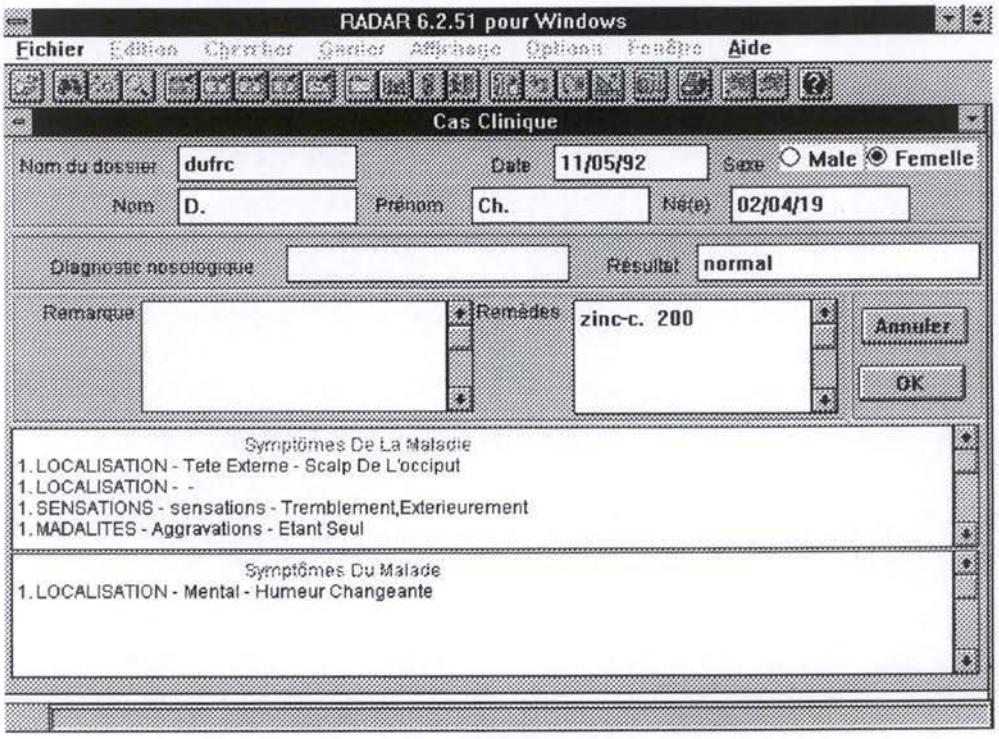


Figure 7-19 Edition du cas clinique

7.2.4 REPERTORISTION AVEC LE SYNTHESIS

Dans ce qui suit, nous avons essayé de dresser une répertorisation du cas présent, sur base du SYNTHESIS.

7.2.4.1 LA COLLECTE DE SYMPTOMES

Les symptômes considérés, comme l'illustre la *figure 6-20*, traduisent bien le cas du patient, et sont pratiquement les mêmes que ceux tirés du répertoire de Boenninghausen, sauf bien évidemment, qu'ils viennent du SYNTHESIS.

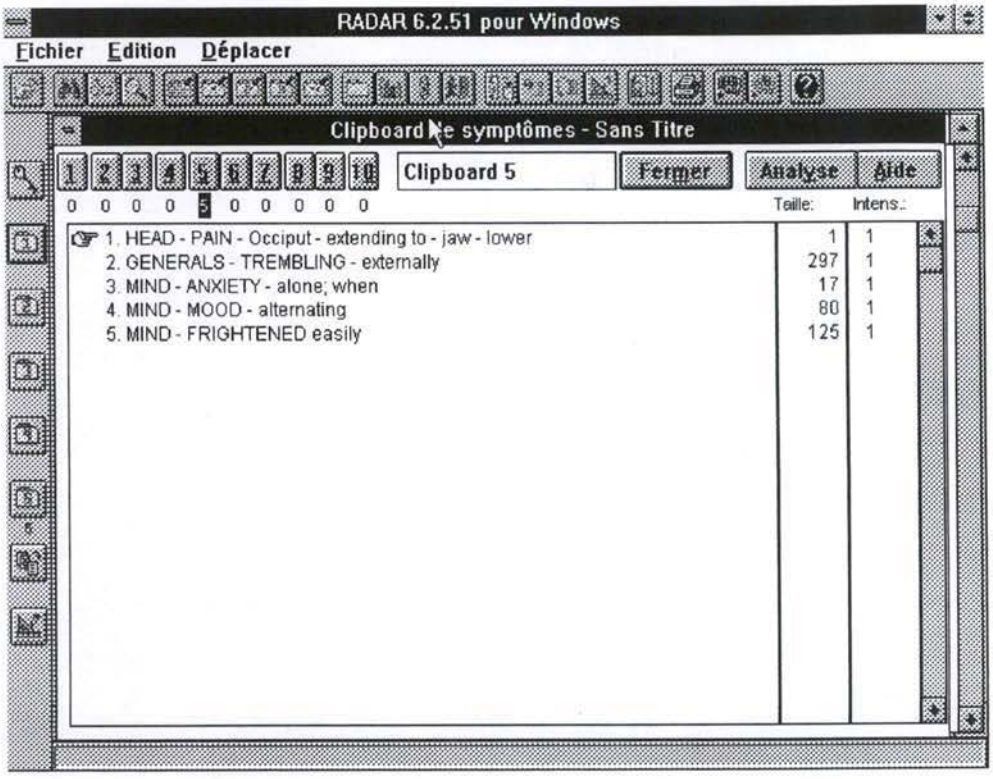


Figure 7-20 Répertorisation avec le SYNTHESIS

7.2.4.2 ANALYSE DU CAS

L'examen de la*figure 6-21* montre que:

- ◊ Arsenicum couvre 4 symptômes sur avec un poids total de 10.
- ◊ Phosphorus couvre 4 symptômes sur 5 avec un poids de 9.
- ◊ Kalium carbonicum couvre 4 symptômes sur 5 avec un poids de 8
- ◊ Zinc a un poids total de 8 et couvre 4 symptômes soit 5.

Dans cette dernière analyse, il faut forcément poser un diagnostic remédial. Car à priori, rien ne nous permet de choisir Zinc qui pourtant s'est révélé efficace pour la patiente.

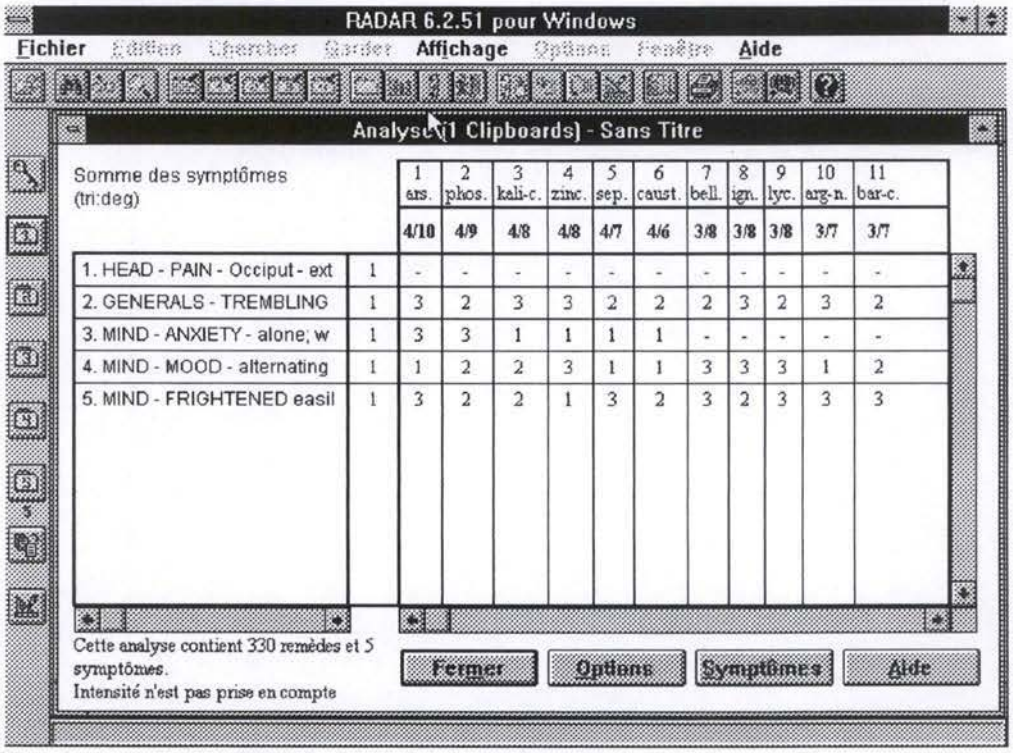


Figure 7-21 Analyse du cas

Conclusions

8. CONCLUSION

De ces deux études de cas, nous tirons deux choses.

Nous avons développé un outil certes répondant à la méthode de Boenninghausen. Mais cet outil ne s'applique pas uniquement au répertoire de Boenninghausen. Dans l'étude de cas 1, notre support était le SYNTHESIS, et les résultats sont plus que satisfaisants, voire même plus immédiats. Boenninghausen nous a laissé une méthode assez synthétique qui recèle tous les fondements de l'homéopathie. Quoi de plus "naturel", de considérer qu'une maladie se caractérise par une localisation, une sensation, une modalité, et des symptômes concomitants qui traduisent l'état du malade. Contrairement à d'autres méthodes qui consistent à associer une intensité plus ou moins importante selon le degré de caractéristique du symptôme, chez Boenninghausen, il n'y a pas de symptômes qui prédominent sur un autre, des symptômes banals deviennent caractéristiques par leur regroupement. Enfin, cet outil ouvre une nouvelle voie qui permettra au médecin homéopathe d'explorer l'univers Boenninghausen avec les répertoires de son choix.

Dans l'étude menée sur le cas 2, nous avons crée deux répertorisations: une sur base du répertoire de Boenninghausen, et une deuxième faite à partir du SYNTHESIS. Apparemment, le répertoire de Boenninghausen semblait mieux indiqué pour le choix du remède. Cependant sur simple base d'une seule étude cas, nous ne sommes pas en mesure de tirer des conclusions conséquentes quant à la validité du répertoire de Boenninghausen par rapport à tout autre répertoire. Néanmois "Rendons à Boenninghausen ce qui revient à Boenninghausen". En effet s'il a fait preuve d'un esprit aussi logique et synthétique lors de la conception de son répertoire comme sa méthode nous le laisse penser, ce qui vraisemblablement est le cas, nous pouvons sans conteste prévaloir ses travaux. Nous citerons ici les travaux remarquables effectués par le docteur Coquillart sur ce sujet. En effet, parmi ces publications, nous lisons pas mal d'études de cas menées qui nous montrent comment Boenninghausen, sa méthode et son répertoire sont déterminants quant au choix du remède final. Le lecteur intéressé se référera à la bibliographie fournie.

Enfin, nous espérons par ce travail lever tous les mites qui tournaient autour de Boenninghausen et de son répertoire. Le médecin homéopathe pourra consulter le répertoire de Boennighausen comme tout autre répertoire, il pourra aussi adopter la méthode de Boenninghausen pour le choix de ses remèdes.

D'un point de vue informatique, ce travail a été enrichissement dans sa conception, de l'analyse à l'implémentation. En outre il nous a permis de mieux nous familiariser avec le langage C, et découvrir un outil qui regorge de potentialités, à savoir le XVT design.

Bibliographie

9. BIBLIOGRAPHIE

[A. JACQUES] A. JACQUES
Introduction à l'homéopathie Hahnemanienne
Facultés des thérapeutiques traditionnelles
Namur, Août 1983

[G. COQUILLART] Guy COQUILLART
Boenninghausen l'entrée ou le désert !
Revue Belge 1995

[G. COQUILLART] G. COQUILLART
Un homme de loi qui est devenu homéopathe.

[DETH] Thorwald DETHLEFSEN
Le destin, une chance à saisir
Radin, 1982

[J. FICHEFET] Jean FICHEFET
Aide à la décision
(syllabus du cours de troisième licence en
Informatique, FUNDP Namur)

[J. FICHEFET] J. FICHEFET, A. JACQUES, P. GARDIN, J. PARIS
RADAR (un système expert à base de
connaissance pour l'homéopathie)

[GUINEE] Roberts GUINEE
De la valeur des symptômes
Exposé congrès homéopathique ALPHEUSDAL

[GREGOIRE] Jean-Claude GREGOIRE
Le répertoire de Boenninghausen, étude historique et analyse de la méthode

[IBET] Quelle est notre approche personnelle du patient
Réflexions menées à propos des symptômes concomitants
Groupes Homéopathes Belge

[RADAR] RADAR user's manual (version 6.1)

[VITH] Georges VITHOULKAS
La science de l'homéopathie
Editions du Rocher, 1984

[VITH2] Georges VILTHOULKAS
L'homéopathie origines et avenir d'une nouvelle médecine
Payot, 1981

Annexe

Listings

A. ANNEXES

A.1 FICHIERS DE DECLARATIONS

A.1.1 FICHIER DE DECLARATION POUR LE MODULE VBOEN

```
/*****************************************************************************

        SOURCE FILE
                VBOEN.H

        MODULE
                Display of the Symptom in memory

        AUTHOR
                Diagne Papa Moussa

        DESCRIPTION

                IMPORTANT: This include file implies an include to XVT.
                ------------------------------------------------------

*****************************************************************************/


/*---- INCLUDE SECTION ----------------------------------------------------*/

#ifndef EXTERNAL
#define  EXTERNAL    extern
#endif

/*---- DEFINE SECTION -----------------------------------------------------*/

#define    VBOEN_SYMPTOM_INTERLIGNE              0

/* 7005 - 7006 */
#define    VBOEN_OFFSET_FOR_SECOND_LANG         10


#define    VBOEN_REFRESH                         1
#define    VBOEN_PAGE_DOWN                       2
#define    VBOEN_PAGE_UP                         3
#define    VBOEN_LINE_DOWN                       4
#define    VBOEN_LINE_UP                         5
#define    VBOEN_GO_TO_FIRST                     6
#define    VBOEN_GO_TO_LAST                      7
#define    VBOEN_DUMMY_REFRESH                   8


#define    OBJECT_VBOEN_SYMPTOM                  1
#define    OBJECT_VBOEN_CHAPTER                  2
#define    OBJECT_VBOEN_REMEDY                   3



#define    VBOEN_KEEP                            1
#define    VBOEN_NO_KEEP                         2

#define    VBOEN_KEEP_OBJECT                     1
#define    VBOEN_KEEP_SELECT                     2

#define    VBOEN_KEEP_DISPLAY                    1
#define    VBOEN_KEEP_NO_DISPLAY                 2

#define    ON                                    1
#define    OFF                                   0
/*---- TYPEDEF SECTION ----------------------------------------------------*/

typedef struct {
         unsigned short     Type;
         unsigned long      Id;
         RCT                Rect;
         void               *Next;
#ifdef DEBUG
         unsigned char     Check;
#endif
```

```
} VBOEN_OBJECT;

typedef struct {
        int             Button;
        RCT             Rect;
#ifdef DEBUG
        unsigned char   Check;
#endif
} RECT_BUTTON;


typedef struct {
        int         Type;
        WINDOW      Win;
        WINDOW      ModifySymptomWindow;
        RCT         SymptomRect,
                       ChapterRect,
                       NbRemediesRect;
        int         FirstSpt,
                       LastSpt,
                       OnlyOneScreen,
                       FirstDisplay,
                       Column,
                       SCNumber,
                       StartAnalysis,
                       CurrentSpt;
        long        Timer;
        VBOEN_OBJECT *Object,
                       *Select;
        int         Focus;
        REPERTORY_WINDOW *Rep;
} VBOEN_WINDOW;

/*---- VARIABLE SECTION -------------------------------------------------*/

/*---- FUNCTION PROTOTYPES ----------------------------------------------*/

EXTERNAL  void          InitializeViewBoenWindow( REPERTORY_WINDOW *, int );
EXTERNAL  void          TerminateViewBoenWindow ( WINDOW );
EXTERNAL VBOEN_OBJECT  *GetViewBoenObjectInList ( VBOEN_OBJECT *,VBOEN_OBJECT*);
EXTERNAL  void          FreeViewBoenObject      ( int, int );

EXTERNAL  void          DisplayViewBoenSymptom  ( WINDOW );
EXTERNAL  void          DisplayViewBoenWindow   ( WINDOW, int );
EXTERNAL  void          DisplayViewBoenSelect   ( WINDOW, VBOEN_OBJECT *, int );

EXTERNAL  void          KeepViewBoenSelect   ( int, PNT *, PNT *, unsigned long, int );
EXTERNAL  void          KeepViewBoenObject (int, PNT *, PNT *, unsigned long, int,int);

EXTERNAL  void          ViewBoenKeyboardHandler   ( WINDOW, EVENT * );
EXTERNAL  void          ManageViewBoenUserEvent   ( WINDOW, EVENT * );
EXTERNAL  void          ViewBoenMouseHandler      ( WINDOW, EVENT * );
EXTERNAL  void          ChangeViewBoenClipboard   ( WINDOW, int );

EXTERNAL  VBOEN_OBJECT  *GetViewBoenObjectAtPoint ( PNT *, int );


EXTERNAL  void          SetViewBoenCurrentSymptom ( WINDOW );
EXTERNAL  void          ChangeViewBoenLine        ( WINDOW, int);

EXTERNAL  int           CopySptToSC                ( int,int, SYMPTOM_IN_MEMORY *);


EXTERNAL  int           GetSymptomTextToDisplay   ( SYMPTOM_IN_MEMORY * );
EXTERNAL  void          DisplayVBoenButton         ( int );
EXTERNAL  void          ManageVBoenButton          ( PNT *, int);
EXTERNAL  void          ManageVBoenKeyboardButton ( int _i );
EXTERNAL  void          ManageCheckButton          ( PNT *, int);
EXTERNAL  void          ManageCheckKeyboardButton (int _i);
EXTERNAL  void          RefreshButton              ( int , int );
EXTERNAL  void          RefreshSelectSymptom        ();
```

A.1.2 FICHIER DE DECLARATION POUR LE MODULE CHECK

```
/*****************************************************************************

        SOURCE FILE
              CHECK.H
```

```
        MODULE
                check complete symptom

        AUTHOR
                DIAGNE Papa Moussa                      26/09/96

        DESCRIPTION

                IMPORTANT: This include file implies an include to XVT.
                ------------------------------------------------------

*****************************************************************************/


/*---- INCLUDE SECTION ----------------------------------------------------*/

#ifndef EXTERNAL
#define  EXTERNAL   extern
#endif

/*---- DEFINE SECTION -----------------------------------------------------*/


/*---- TYPEDEF SECTION ----------------------------------------------------*/

typedef struct {
          int               Type;
          WINDOW            Win;
          int               Focus;
          int               SCNumber;
          int               DisplayNumber;
} CHECK_WINDOW;


typedef struct {
                unsigned char Present;
                unsigned char Complet;
                unsigned char Check;
                } COMPLET_SYMPTOM;



/*---- VARIABLE SECTION ---------------------------------------------------*/


/*---- FUNCTION PROTOTYPES ------------------------------------------------*/

EXTERNAL  void              InitializeViewCheckWindow ( );
EXTERNAL  int               DisplayCheckWindow         (int);
EXTERNAL  void              InitCheck                 ();
EXTERNAL  unsigned char     CheckType                 ( int );
EXTERNAL  void              CheckInMemory             ( int );
EXTERNAL  void              CheckCompleteSymptom      ();

EXTERNAL  void              ViderSpt                  ();
EXTERNAL  void              RemplirSpt                ();
EXTERNAL  void              VboenAnalyse              ();
EXTERNAL  void              NewVboenAnalyse           ();
EXTERNAL  int               CopySptToSC_5             (SYMPTOM_IN_MEMORY *);
EXTERNAL  int               GoToAnalyse               ( );
EXTERNAL  void              DisplayCompleteSymptom    ( int );
EXTERNAL  void              UpdateCheckWindow         ( WINDOW );
```

A.1.3 FICHIER DECLARATION DU MODULECLINICAL

```
/*****************************************************************************

        SOURCE FILE
                VCLIN.H

        MODULE
                Manage the clinical view

        AUTHOR
                Papa Moussa Diagne

        DESCRIPTION

                IMPORTANT: This include file implies an include to XVT.
```

```
                ------------------------------------------------------
*****************************************************************************/


/*---- INCLUDE SECTION ----------------------------------------------------*/

#ifndef EXTERNAL
#define  EXTERNAL    extern
#endif

/*---- DEFINE SECTION -----------------------------------------------------*/

#define    CLIN_REFRESH_1                    1
#define    CLIN_PAGE_DOWN_1                  2
#define    CLIN_PAGE_UP_1                    3
#define    CLIN_LINE_DOWN_1                  4
#define    CLIN_LINE_UP_1                    5
#define    CLIN_DUMMY_REFRESH_1              6

#define    CLIN_REFRESH_2                    7
#define    CLIN_PAGE_DOWN_2                  8
#define    CLIN_PAGE_UP_2                    9
#define    CLIN_LINE_DOWN_2                  10
#define    CLIN_LINE_UP_2                    11
#define    CLIN_DUMMY_REFRESH_2              12


/*---- TYPEDEF SECTION ----------------------------------------------------*/

typedef struct {
         int              Type,
                          Focus,
                          Line1,
                          Line2;
         WINDOW           Win;
         RCT              MaladieRect,
                          MaladeRect;
         int              FirstSpt_1,
                          LastSpt_1,
                          CurrentSpt_1,
                          FirstSpt_2,
                          LastSpt_2,
                          CurrentSpt_2;
} CLIN_WINDOW;


/*---- VARIABLE SECTION ---------------------------------------------------*/


/*---- FUNCTION PROTOTYPES ------------------------------------------------*/

EXTERNAL  void              DisplayViewMaladieSymptom   (WINDOW,int);
EXTERNAL  void          UpdateViewCliniqueWindow   (WINDOW);
EXTERNAL  void          UpdateNumberWindow         (WINDOW);
EXTERNAL  void          UpdatePatientCase          ();

EXTERNAL  void          DisplayViewMaladeSymptom   (WINDOW,int);
EXTERNAL  void          InitializeViewCliniqueWindow ();
EXTERNAL  void          CalculateSymptomMaladie      ();
EXTERNAL  void          CalculateSymptomMalade       ();
EXTERNAL  void          CalculateSymptomCase         ();
EXTERNAL  int           CopySptToClinique     (int, unsigned char, SYMPTOM_IN_MEMORY * );
EXTERNAL  int           CopySptToQualif   (int,int,unsigned char, SYMPTOM_IN_MEMORY * );
EXTERNAL  int           CopySptToDegrees  (int,int,unsigned char, SYMPTOM_IN_MEMORY * );
```

A.1.4 FICHIER DECLARATION DU MODULE VALORISATION

```
/*****************************************************************************/

                SOURCE FILE
                                TAKE.H

                MODULE
                                Valorize symptoms management

                AUTHOR
                               Diagne Papa Moussa

                DESCRIPTION
                                This include file define all structures and variables used
                                to valorize symptoms .
```

```
                                    IMPORTANT: This include file implies an include to XVT.
                                    -------------------------------------------------------

*****************************************************************************/

/*---- INCLUDE SECTION ----------------------------------------------------*/

#ifndef EXTERNAL
#define  EXTERNAL   extern
#endif

/*---- DEFINE SECTION -----------------------------------------------------*/

#define   LOCALISATION 1
#define   SENSATION    2
#define   MODALITE     4
#define   CONCOMITTANT 8


/*---- TYPEDEF SECTION ----------------------------------------------------*/


typedef struct {
       unsigned short   RemedId;
       unsigned char    Degree;
       unsigned short   AuthorId;
/*   unsigned short   Views;*/ /*6005*/
} REMED_LIST;


typedef struct S {
       unsigned char    *SptTextLg1;
       unsigned char    *SptTextLg2;
       char             Intensity;
       unsigned char    Group;
       REMED_LIST       *RmdList;     /* Pointer to a table of Remedy List */
       unsigned char    Degrees;
       unsigned char    Qualif;
       unsigned short   NbRemedies;
       char             Langue[2];
       unsigned char    Origin;
       unsigned short   View; /*6006*/
       struct S             *Next;
} SYMPTOM_IN_MEMORY;


typedef struct {
       char                 InMemory;
       unsigned short       NbSpt;
       char                 *Name;
       SYMPTOM_IN_MEMORY    *First;
       SYMPTOM_IN_MEMORY    *Last;
} SYMPTOM_CLIPBOARD;


/*---- VARIABLE SECTION ---------------------------------------------------*/

/*---- FUNCTION PROTOTYPES ------------------------------------------------*/
EXTERNAL  void            BDeleteSptInMemory           ( int, int );
EXTERNAL  void            BDeleteSymptoms              ( );

EXTERNAL  void            GetNumberInMemory           ( int , int, int );
EXTERNAL  void            GetNumberSymptom            ( int );

EXTERNAL  void            DeleteCurrentNumberInMemory  ( int, int,int);
EXTERNAL  void            DeleteCurrentNumberSymptom   ( int );

EXTERNAL  void            DeleteNumberInMemory         ( int, int);
EXTERNAL  void            DeleteNumberSymptom          ( );
```

A.2 SOURCE MODULE

A.2.1 MODULE VBOEN

```
/*****************************************************************************/
       SOURCE FILE
```

```
                VBOEN.C

        MODULE
                View Taken Symptoms window

        AUTHOR
              Diagne Papa Moussa

        DESCRIPTION
                Contain all functions for the management of the
                View Taken Symptoms window
*****************************************************************************/

/*---- INCLUDE SECTION ----------------------------------------------------*/

#include "xvt.h"
#include "x_radar.h"
#include "radar.h"
#include "math.h"

/*---- DEFINE SECTION -----------------------------------------------------*/


/*---- VARIABLE SECTION ---------------------------------------------------*/

static WIN_MSG WinViewBoenMsg [] = {
                W_VBOEN,                621, '\0',
                W_VBOEN_OK,             622, '\0',
                W_VBOEN_ANALYS,         623, '\0',
                W_VBOEN_HELP,           624, '\0',
                W_VBOEN_SC_LOC,         4000, '\0',
                W_VBOEN_SC_SENS,        4001, '\0',
                W_VBOEN_SC_MOD,         4002, '\0',
                W_VBOEN_SC_CONC,        4003, '\0',
                /*W_VBOEN_SC_SEL_ALL,     4004, '\0',
                W_VBOEN_SC_6,           2522, '\0',
                W_VBOEN_SC_7,           2523, '\0',
                W_VBOEN_SC_8,           2524, '\0',
                W_VBOEN_SC_9,           2525, '\0',
                W_VBOEN_SC_10,          2526, '\0',*/
                0,                          0, '\0'
};

/* VBOEN MENU MESSAGE DEFINITION */

#define     SBM_NB_MENUS        3

#define     SBM_FILE_ITEMS      6
#define     SBM_EDIT_ITEMS      7
#define     SBM_MOVE_ITEMS      4

#define     SBM_MAX_NB_ITEMS    SBM_EDIT_ITEMS

short     SBMenusMsgNumbers [SBM_NB_MENUS] = {
       490, 491, 971
};

short      SBItemsMsgNumbers [SBM_NB_MENUS] [SBM_MAX_NB_ITEMS] = {
/* file menu */
       3192,3193,   0,1219,1220,1221,   0,
/* edit menu */
        4020,   0,4005,4006,4007,   0, 4008,
/* move menu */
        972, 975, 973, 974,   0,   0,   0,
};

/*---- LOW LEVEL FUNCTIONS PROTOTYPES -------------------------------------*/


/*---- HIGH FUNCTION SECTION ----------------------------------------------*/


/*****************************************************************************

  NAME
                InitializeViewBoenWindow - Initialize the ViewBoen window and structures

  SYNTAX
                void
                InitializeViewBoenWindow ( rep, clipboard )

  PARAMETERS
                REPERTORY_WINDOW    *rep;            I  Repertory window
                int                 clipboard;       I  CLipboard to view first
                int                 execute_analyse; I  1=Start Analysis at window
```

```
                        close.

                      0=Don't start Analysis.

  INCLUDE
                xvt.h
                radar.h

  DESCRIPTION
                This function makes all initializations for the ViewBoen window

                1 - Create the ViewBoen window

                2 - Allocates the ViewBoen structure

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************************/

void
InitializeViewBoenWindow ( rep, clipboard )
REPERTORY_WINDOW *rep;
int              clipboard;
{
WINDOW  _vboen;
char    _buf [100];
WIN_DEF *_windef;
RCT     *_prct;

        xvt_win_set_cursor ( TASK_WIN, CURSOR_WAIT );
        DbVirtualClose ( 2 );

/*
! 7065 We must remove any opened ViewBoen Window
! 7064
*/

        if ( WinViewBoen != NULL ) TerminateViewBoenWindow ( WinViewBoen->Win );


        if ( ! ( WinViewBoen = (VBOEN_WINDOW *)Malloc ( sizeof ( VBOEN_WINDOW ) + 1 ) ) ) {
                xvt_dm_post_note( GetMessage ( FileMessage, 104, Option.General.Lang, &RecMessage) );
                xvt_win_set_cursor ( TASK_WIN, CURSOR_ARROW );
                return;
        }


        WinViewBoen->Type            = WINDOW_TYPE_VBOEN;
        WinViewBoen->Rep             = rep;

        WinViewBoen->Object              = (VBOEN_OBJECT *)NULL;
        WinViewBoen->Select              = (VBOEN_OBJECT *)NULL;
        WinViewBoen->FirstSpt            = (int) 0;
        WinViewBoen->SCNumber            = clipboard;
        WinViewBoen->LastSpt             = 0;
        WinViewBoen->OnlyOneScreen       = TRUE;
        WinViewBoen->FirstDisplay        = TRUE;
        WinViewBoen->Column              = 1;
        WinViewBoen->StartAnalysis       = 0;

        DbVirtualClose ( 2 );

         /* Create and calculate window */

         if ( ! ( _windef = xvt_res_get_win_def ( W_VBOEN ) ) ) {
                xvt_dm_post_note( GetMessage ( FileMessage, 31, Option.Gene ral.Lang, &RecMessage ) );
                Free ( (char *)WinViewBoen );
                xvt_win_set_cursor ( TASK_WIN, CURSOR_ARROW );
                return;
         }

         SetWindowRectangle ( _windef, WINDOW_VERT );
         if ( _prct = GetControlRectangle ( _windef, W_VBOEN_VSCROLL ) ) {
                _prct->bottom = _windef [ 0 ].rct.bottom - _windef [ 0 ].rct.top - 4;
         }

         if ( !( _vboen = xvt_win_create_def ( _windef, TASK_WIN, EM_ALL,
          W_VBOEN_eh, (long)WinViewBoen ) ) ) {
                xvt_dm_post_note( GetMessage ( FileMessage, 31, Option.General.Lang, &RecMessage )  );
                Free ( (char *)WinViewBoen );
```

```
                xvt_win_set_cursor ( TASK_WIN, CURSOR_ARROW );
                return;
        }
        xvt_res_free_win_def ( _windef );



        WinViewBoen->Win                = _vboen;


        /* Initialize Focus */

        WinViewBoen->Focus              = 0;
        WindowTabHandler ( WinViewBoen->Win, (EVENT *)NULL, WinViewBoen->Focus,
                                        W_VBOEN_OK, W_VBOEN_VSCROLL, TRUE, W_VBOEN_OK );

                        /* Set national menu */

        LoadNationalMenus ( WinViewBoen->Win, SBMenusMsgNumbers,
                                        SBItemsMsgNumbers,
                                        SBM_NB_MENUS,
                                        SBM_MAX_NB_ITEMS );

                /* Load National text */

        LoadMessagesForWindow ( WinViewBoen->Win, (WIN_MSG *) WinViewBoenMsg );


        SetWindowScrollBar ( xvt_win_get_ctl ( WinViewBoen->Win, W_VBOEN_VSCROLL ),
                                HVSCROLL, 1L, (long)10, 1L );

        /* On met le nom du cas dans le nom de la fen^tre */
        xvt_vobj_get_title ( _vboen, _buf, 100 );
        strcat ( _buf, " - " );
        strcat ( _buf, CurrentCaseInMemory );

        xvt_vobj_set_title ( _vboen, _buf );


        xvt_win_set_cursor ( TASK_WIN, CURSOR_ARROW );
}


/*****************************************************************************

  NAME
                TerminateViewBoenWindow - Terminate the ViewBoen window and structures

  SYNTAX
                void
                TerminateViewBoenWindow ( win )

  PARAMETERS
                WINDOW          win;        I   ViewBoen window

  INCLUDE
                xvt.h
                radar.h

  DESCRIPTION
                This function makes all close for the ViewBoen window

                1 - Free the structures

                2 - Close the window

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************************/

void
TerminateViewBoenWindow ( win )
        WINDOW    win;
{
int  _i;

        if ( WinViewBoen->ModifySymptom == TRUE ) {
                xvt_vobj_destroy ( WinViewBoen->ModifySymptomWindow );
        }
```

```
        if ( WinSaveCase != ( SAVECASE_WINDOW * ) NULL )  {
                TerminateSaveCase ( WinSaveCase->Win );
        }

        _i = WinViewBoen->StartAnalysis;
        FreeViewBoenObject ( VBOEN_KEEP_OBJECT, VBOEN_KEEP_NO_DISPLAY );
        FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_NO_DISPLAY );
        Free ( (unsigned char *)WinViewBoen );

        WinViewBoen = NULL;

        xvt_vobj_destroy ( win );


        if ( WinVIndex ) {
                xvt_vobj_raise( WinVIndex->Win );
        }
        else {
                if ( OldWindow ) {
                        xvt_vobj_raise( OldWindow );
                        xvt_dwin_invalidate_rect ( OldWindow, NULL );
                }
        }

        if ( _i ) {
                InitializeVboenAnalyse ( 1 );
        }

}

/*****************************************************************************

  NAME
                FreeViewBoenObject - Free the ViewBoen object list

  SYNTAX
                void
                FreeViewBoenObject ( keep, display )

  PARAMETERS
                int             keep,       I   Type of keep, object or selected objet
                                                    VBOEN_KEEP_OBJECT or
                                                    VBOEN_KEEP_SELECT
               display;     I   Before free, have to display object ?
                                                    VBOEN_KEEP_DISPLAY or
                                                    VBOEN_KEEP_NO_DISPLAY.

  INCLUDE
                xvt.h
                radar.h

  DESCRIPTION
                This function frees the list of object position and reset the pointer
                to a null value.

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************************/

void
FreeViewBoenObject ( keep, display )
        int                 keep,
                           display;
{
VBOEN_OBJECT     *_ptr, *new_pos;

        if ( WinViewBoen ) {

                if ( keep == VBOEN_KEEP_OBJECT ) {
                        while ( WinViewBoen->Object ) {
                                _ptr = WinViewBoen->Object;
                                WinViewBoen->Object = _ptr->Next;
                                Free ( (char *)_ptr );
                        }
                }
                else {
                         while ( WinViewBoen->Select ) {
```

```
                                _ptr = WinViewBoen->Select;

                                /* If display needed and if this object
                                        is displayed on this current screen
                                        redisplay object in normal mode
                                */

                                WinViewBoen->Select = _ptr->Next;
                                if ( display == VBOEN_KEEP_DISPLAY ) {
                                        new_pos = GetViewBoenObjectInList ( WinViewBoen->Object, _ptr );
                                        if ( new_pos ) {
                                                if ( new_pos->Id == _ptr->Id ) {
                                                        DisplayViewBoenSelect ( WinViewBoen->Win, new_pos,
                                                        ATTRIBUTE_NORMAL );}
                                                else {
                                                        DisplayViewBoenSelect ( WinViewBoen->Win, _ptr,
                                                        ATTRIBUTE_NORMAL );
                                                }
                                        }
                                }
                                Free ( (char *)_ptr );
                        }


                }
        }
}

/*****************************************************************************

  NAME
                GetViewBoenObjectInList - Search for an object in an object list

  SYNTAX
                VBOEN_OBJECT
                *GetViewBoenObjectInList ( list, object )

  PARAMETERS
                VBOEN_OBJECT  *list,   I   List of object where to search
                              *object; I   Object to search for

  INCLUDE
                xvt.h
                radar.h

  DESCRIPTION
                This function search in an object list for an specific object.

  RETURN
                Pointer to the found object or NULL if no object in this list.

  EXAMPLE

  SEE ALSO

*****************************************************************************/

VBOEN_OBJECT
*GetViewBoenObjectInList ( list, object )
        VBOEN_OBJECT    *list,
                        *object;
{
 VBOEN_OBJECT   *_vobj;

        _vobj = list;
        while ( _vobj && ! ( _vobj->Type == object->Type && _vobj->Id == object->Id ) ) {
                _vobj = _vobj->Next;
        }

        return ( _vobj );
}

/*****************************************************************************

  NAME
                DisplayViewBoenWindow - Display the symptom list

  SYNTAX
                int
                DisplayViewBoenWindow ( win, mode )

  PARAMETERS
                WINDOW          win;        I   ViewBoen window
                int             mode;       I   Mode for display
                                VBOEN_REFRESH
```

```
                                VBOEN_PAGE_UP
                                VBOEN_PAGE_DOWN
                                VBOEN_LINE_UP
                                VBOEN_LINE_DOWN

  INCLUDE
                xvt.h
                radar.h

  DESCRIPTION
                This function display the list of symptoms for the selected
                clipboard.

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************/
void
DisplayViewBoenWindow ( win, mode )
        WINDOW    win;
        int       mode;
{
SYMPTOM_IN_MEMORY  *SptInMemory;
VBOEN_OBJECT    _vobj,
                *_selected;
RCT               _rct_spt_list,
                  _rct,
                  _rct1,
                  _rct_chp_list,
                  _rct_rmd_list;
PNT               _from,
                  _to,
                  _from2;
int               _leading2,
                  _ascent2,
                  _descent2;
short             _leading,
                  _ascent,
                  _descent,
                  _size,
                  _top,
                  _size_for_spt_nb,
                  _next_verti;
int               _st,
                  _width,
                  _attrib,
                  _attrib_spt,
                  _col,
                  _i,
                  _nblg,
                  _j,
                  _lang    [ 2 ],
                  _typetxt [ 2 ],
                  _typetext;
double            _x, _y;
unsigned char     *_text [ 2 ],c1;
int               num;
char              _buf [20];

FONT /* XVT_FNTID xvt_R3FNT */                Font_For_Spt_Number;


        xvt_win_set_cursor ( TASK_WIN, CURSOR_WAIT );


        switch ( mode ) {
                case VBOEN_DUMMY_REFRESH:
                        FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_NO_DISPLAY );
                        break;


                case VBOEN_REFRESH:
                        if ( WinViewBoen->Object == (VBOEN_OBJECT *) NULL ) {
                                WinViewBoen->FirstSpt = 0;
                                WinViewBoen->CurrentSpt = 0;
                        }
                        for ( _i = 0; _i <16; _i++ )
                           {
                            RectButton[_i].Button = ON;
                            DisplayVBoenButton ( _i);
                            }
```

```
                        break;

                case VBOEN_LINE_DOWN:
                        break;

                case VBOEN_PAGE_DOWN:
                        if (WinViewBoen->LastSpt < (int)SC [WinViewBoen->SCNumber].NbSpt) {
                                WinViewBoen->FirstSpt = WinViewBoen->LastSpt;
                                WinViewBoen->CurrentSpt = WinViewBoen->FirstSpt;
                        }
                        break;

                case VBOEN_LINE_UP:
                        WinViewBoen->FirstSpt --;
                        if ( WinViewBoen->FirstSpt < 0) WinViewBoen->FirstSpt = 0;
                        WinViewBoen->CurrentSpt = WinViewBoen->FirstSpt;
                        break;

                case VBOEN_PAGE_UP:
                        WinViewBoen->FirstSpt -= 7;
                        if ( WinViewBoen->FirstSpt < 0) WinViewBoen->FirstSpt = 0;
                        WinViewBoen->CurrentSpt = WinViewBoen->FirstSpt;
                        break;

                case VBOEN_GO_TO_FIRST:
                        WinViewBoen->FirstSpt = 0;
                        WinViewBoen->CurrentSpt = 0;

                        break;

                case VBOEN_GO_TO_LAST:
                        WinViewBoen->FirstSpt = SC [WinViewBoen->SCNumber].NbSpt-1;
                        WinViewBoen->CurrentSpt = WinViewBoen->FirstSpt;
                        break;
        }


        _i  = WinViewBoen->FirstSpt;
        SptInMemory = SC [WinViewBoen->SCNumber].First;
        for (_j=0; _j < _i; _j++) SptInMemory = SptInMemory->Next;

        FreeViewBoenObject ( VBOEN_KEEP_OBJECT, VBOEN_KEEP_NO_DISPLAY );

                        /************************************/
                        /* Affichage rectangle des symptomes */
                        /************************************/

        xvt_vobj_get_outer_rect ( xvt_win_get_ctl ( win, W_VBOEN_VSCROLL ), &_rct1 );
        xvt_rect_set ( &WinViewBoen->SymptomRect,54, _rct1.top, _rct1.left, _rct1.bottom );
        _size = ( WinViewBoen->SymptomRect.right - WinViewBoen->SymptomRect.left ) / 12;
        WinViewBoen->SymptomRect.right -= ( _size*2 );

        WindowDrawRect ( win, &WinViewBoen->SymptomRect, (COLOR)xvt_vobj_get_attr ( win,
                        ATTR_BACK_COLOR ) );
        WindowDrawEmptyRect ( win, &WinViewBoen->SymptomRect, COLOR_BLACK );

        xvt_rect_set ( &_rct_spt_list, WinViewBoen->SymptomRect.left + 25,
                                       WinViewBoen->SymptomRect.top + 2,
                                       WinViewBoen->SymptomRect.right - 2,
                                       WinViewBoen->SymptomRect.bottom - 2);
    /************************************/
    /* Affichage rectangle des chapitres */
    /************************************/

      xvt_rect_set ( &WinViewBoen->ChapterRect,  WinViewBoen->SymptomRect.right,
                                                 WinViewBoen->SymptomRect.top,
                                                 WinViewBoen->SymptomRect.right + _size,
                                                 WinViewBoen->SymptomRect.bottom );
      WindowDrawRect ( win, &WinViewBoen->ChapterRect, (COLOR)xvt_vobj_get_attr ( win,
                              ATTR_BACK_COLOR ) );
      WindowDrawEmptyRect ( win, &WinViewBoen->ChapterRect, COLOR_BLACK );

      xvt_rect_set ( &_rct_chp_list, WinViewBoen->ChapterRect.left + 2,
                                     WinViewBoen->ChapterRect.top + 2,
                                     WinViewBoen->ChapterRect.right - 2,
                                     WinViewBoen->ChapterRect.bottom - 2);


/************************************/
      /* Affichage des remedes */
/************************************/

      xvt_rect_set ( &WinViewBoen->NbRemediesRect, WinViewBoen->ChapterRect.right,
                                                   WinViewBoen->ChapterRect.top,
                                                   WinViewBoen->ChapterRect.right + _size,
```

```
                                                WinViewBoen->ChapterRect.bottom );
WindowDrawRect ( win, &WinViewBoen->NbRemediesRect, (COLOR)xvt_vo bj_get_attr ( win,
                ATTR_BACK_COLOR ) );
WindowDrawEmptyRect ( win, &WinViewBoen->NbRemediesRect, COLOR_BLACK );

xvt_rect_set ( &_rct_rmd_list, WinViewBoen->NbRemediesRect.left + 2,
                               WinViewBoen->NbRemediesRect.top + 2,
                               WinViewBoen->NbRemediesRect.right - 2,
                               WinViewBoen->NbRemediesRect.bottom - 2);


_st = TRUE;
if ( _i >= (int)SC[WinViewBoen->SCNumber].NbSpt ) _st = FALSE;

if ( _st ) {
        _col = 0;
        _from.v = _rct_spt_list.top;
        _from.h = _rct_spt_list.left;

        /***************************/
        /* Affichage des symptomes */
        /***************************/

        while ( _st ) {
                _attrib       = ATTRIBUTE_NORMAL;
                _attrib_spt   = ATTRIBUTE_NORMAL;
                _selected = (VBOEN_OBJECT *)NULL;

                if ( WinViewBoen->Select ) {
                        _vobj.Type = OBJECT_VBOEN_SYMPTOM;
                        _vobj.Id   = (long) _i;

                        if ( _selected = GetViewBoenObjectInList ( WinViewBoen->Select, &_vobj )
                                ) {
                                _attrib_spt = ATTRIBUTE_REVERSE;
                        }
                        else {
                                _attrib_spt = ATTRIBUTE_NORMAL;
                        }
                }

                _lang [ 0 ] = _lang [ 1 ] = -1;


                _text    [ 0 ] = SptInMemory->SptTextLg1;
                if ( SptInMemory->Langue [0] == LANG_UNDEFINED ) {
                        _lang [0] = -1;
                }
                else {
                        _lang [ 0 ] = SptInMemory->Langue[0];
                }
                _lang [ 1 ] = SptInMemory->Langue[1];
                _typetxt [ 0 ] = OBJECT_GENERAL_SYMPTOM_LANG_1;

                /*if ( (SptInMemory->Qualif & (unsigned char) 1) &&
                          (SptInMemory->Qualif & (unsigned char) 2)     ) {
                                  _typetxt [ 0 ] = OBJECT_SYMPTOM_CAUS_ELIM_LANG_1;
                }
                else {
                        if ( SptInMemory->Qualif & (unsigned char) 1)
                                _typetxt [ 0 ] = OBJECT_SYMPTOM_ELIM_LANG_1;
                        else {
                                if ( SptInMemory->Qualif & (unsigned char) 2)
                                        _typetxt [ 0 ] = OBJECT_SYMPTOM_CAUS_LANG_1;
                                else
                                        _typetxt [ 0 ] = OBJECT_GENERAL_SYMPTOM_LANG_1;
                        }
                } */


                if ( SptInMemory->Langue [1] != 0 ) {

                        _text    [ 1 ] = SptInMemory->SptTextLg2;
                         _typetxt [ 1 ] = OBJECT_GENERAL_SYMPTOM_LANG_2;
                        /*if ( (SptInMemory->Qualif & (unsigned char) 1) &&
                                  (SptInMemory->Qualif & (unsigned char) 2)     ) {
                                  _typetxt [ 1 ] = OBJECT_SYMPTOM_CAUS_ELIM_LANG_2;
                        }
                        else {
                                if ( SptInMemory->Qualif & (unsigned char) 1)
                                  _typetxt [ 1 ] = OBJECT_SYMPTOM_ELIM_LANG_2;
                                else {
                                        if ( SptInMemory->Qualif & (unsigned char) 2)
                                          _typetxt [ 1 ] = OBJECT_SYMPTOM_CAUS_LANG_2;
                                        else    _typetxt [ 1 ] = OBJECT_GENERAL_SYMPTOM_LANG_2;
```

```
                }
        } */

        /* On regarde quelle langue on doit afficher en fonction du paramŠtre
                dans le RADAR.INI
        */

        if ( GetSymptomTextToDisplay ( SptInMemory ) ) {
                /* Rien … faire les langues sont bonnes */
        }
        else {
                /* On remet l'anglais en premiŠre langue */
                SwapData ( &_text [ 0 ], &_text [ 1 ], sizeof ( _text [ 0 ] ) );
                SwapData ( &_lang [ 0 ], &_lang [ 1 ], sizeof ( _lang [ 0 ] ) );

        }

}
_lang [1] = -1; /* POur eviter d'afficher les deux langues m^me
                                                apres le swap data
                                        */

 /* Calculates needed space for symptom number */
  /*  Si Clipboard 5 on affiche non pas le numero
de symptomes mais plutot le numero du symptome complet,
j'ai Qualif + Degrees, car l'un des deux est toujours nul,
en principe je devais faire un test sur Qualif et Degrees
pour savoir lequel est non nul pour l'afficher*/

if(WinViewBoen->SCNumber == 4)
   {
      c1 = SptInMemory->Qualif;
      for(num =1; c1 !=0; c1>>=1) {
         if (c1 & 1)
         sprintf(_buf,"%d. ", num);
         ++num ;
       }
      c1 = SptInMemory->Degrees;
      for(num =1; c1 !=0; c1>>=1) {
         if (c1 & 1)
         sprintf(_buf,"%d. ", num);
         ++num ;
       }


    }

else
sprintf(_buf, "%d. ", _i+1);
if ( _lang[0] == -1) _typetext = _typetxt[1];
else                 _typetext = _typetxt[0];
_size_for_spt_nb = GetWindowObjectDim ( win, _typetext, _buf,
                                &_leading, &_ascent, &_descent );

 /* Calculate needed space to display */
/* Calcul du nombre de ligne pour la 2eme. langue */
_nblg = 0;
_width = 0;

if (_lang[1] != -1) {
        _width = GetWindowObjectDim ( win, _typetxt[1], _text[1],
                &_leading, &_ascent, &_descent );
        if (_lang[0] == -1) _width += _size_for_spt_nb;
        _y = (float)_width;
        _x = _y / (_rct_spt_list.right - _rct_spt_list.left);

        if (_x < (float)1.0) _nblg = 1;
        else {
                _nblg =  (int)_x + 1;
        }
}

/* Calcul du nombre de ligne pour la 1ere. langue */
if (_lang[0] != -1) {
        _width = GetWindowObjectDim ( win, _typetxt[0], _text[0],
                        &_leading, &_ascent, &_descent );

        _width += _size_for_spt_nb;
        _y = (float)_width;
        _x = _y / (_rct_spt_list.right - _rct_spt_list.left);

        if (_x < (float)1.0) _nblg += 1;
        else {
                _nblg +=  (int)_x ;
                _nblg ++;
```

```
                        }
                }

                _size = (_nblg * (_leading + _ascent + _descent) );
                if ( _from.v + _size >_rct_spt_list.bottom) _st = FALSE;

                _next_verti     = _from.v;
                _from2.h        = _from.h;
                _from2.v        = _from.v;

                for (_j=0; _j<2 && _st; _j++) {
                        /**********************/
                        /* Affichage du texte */
                        /**********************/
                        if ( _st) {
                                if ( _j == 0 ) {
                                        /* Affichage du numero de Spt */
                                        if ( _lang[0] == -1) _typetext = _typetxt[1];
                                        else                 _typetext = _typetxt[0];

                                        DisplayRepertorySentense ( win, _typetext, _buf, "",
                                        FALSE, &_from, &_to, _attrib_spt,
                                        _rct_spt_list.left,
                                        _rct_spt_list.right,
                                        REPERTORY_CANNOT_SPLIT );
                                        _from.h += _size_for_spt_nb;
                                }
                                if ( _j == 1 ) {
                                        _from.h += _size_for_spt_nb;
                                }
                                if ( _j == 1 && _lang[0] != -1 && _lang[1] != -1 ) {
                                        _from.h += VBOEN_OFFSET_FOR_SECOND_LANG;
                                }
                                if (_lang[_j] != -1) {

                                DisplayRepertorySentense ( win, _typetxt[_j], _text[_j], "",
                                                        FALSE, &_from, &_to, _attrib_spt,
                                                        _rct_spt_list.left,
                                                        _rct_spt_list.right,
                                                        REPERTORY_CANNOT_SPLIT );
                                        _next_verti = _to.v;
                                }

                                if ( _j == 1 || _lang [1] == -1 ) {
                                        _to.h = _rct_spt_list.right;
                                        _to.v = _next_verti;


                                        KeepViewBoenObject ( OBJECT_VBOEN_SYMPTOM, &_from2,
                                                                &_to, _i, VBOEN_KEEP_OBJECT,
                                                                VBOEN_KEEP_NO_DISPLAY );
                                }

                                if (_j == 0) {
                                        /**********************************/
                                        /* Affichage du nombre de remedes */
                                        /**********************************/
/*
! 7050 : Counts the number of different remedies
 sprintf(_buf, "%d", SptInMemory->NbRemedies);

*/
                sprintf(_buf, "%d", NumberOfDifferentRemedies (SptInMemory) );
                /*sprintf(_buf, "%c", SptInMemory->Origin);*/
                        _width = GetWindowObjectDim ( win, OBJECT_GENERAL_SYMPTOM_LANG_1, _buf,
                                                &_leading, &_ascent, &_descent );

                        _from.v = _from2.v;
                        _from.h = _rct_rmd_list.right - 3 - _width;
                        DisplayWindowObject ( win, &_from, &_to, OBJECT_GENERAL_SYMPTOM_LANG_1, _buf,
                                        "", _attrib );
                                        _from.h = _rct_rmd_list.left  - 3;
                                        _to.h   = _rct_rmd_list.right + 3;

                        KeepViewBoenObject ( OBJECT_VBOEN_REMEDY, &_from, &_to, _i,
                                        VBOEN_KEEP_OBJECT, VBOEN_KEEP_NO_DISPLAY );

                                        /**********************************/
                                        /* Affichage du titre de symptomes */
                                        /**********************************/

        if(SptInMemory)
          {
            memset(_buf,'\0',20);
            if (SptInMemory->Origin != 'e') sprintf(_buf,"%s", "@");
```

```
            if(WinViewBoen->SCNumber==0)
               strcat (_buf,  GetMessage (FileMessage, 4011, Option.General.Lang, &RecMessage) );
            if(WinViewBoen->SCNumber==1)
               strcat (_buf,  GetMessage (FileMessage, 4012, Option.General.Lang, &RecMessage) );
            if(WinViewBoen->SCNumber==2)
               strcat (_buf,  GetMessage (FileMessage, 4013, Option.General.Lang, &RecMessage) );
            if(WinViewBoen->SCNumber==3)
               strcat (_buf,  GetMessage (FileMessage, 4014, Option.General.Lang, &RecMessage) );
            if(WinViewBoen->SCNumber==4)
               {
                if(SptInMemory->Intensity==0)
                  strcat (_buf,  GetMessage (FileMessage, 4011, Option.General.Lang, &RecMessage) );
                if(SptInMemory->Intensity==1)
                  strcat (_buf,  GetMessage (FileMessage, 4012, Option.General.Lang, &RecMessage) );
                if(SptInMemory->Intensity==2)
                  strcat (_buf,  GetMessage (FileMessage, 4013, Option.General.Lang, &RecMessage) );
                if(SptInMemory->Intensity==3)
                  strcat (_buf,  GetMessage (FileMessage, 4014, Option.General.Lang, &RecMessage) );
               }


         }


 _width = GetWindowObjectDim ( win, OBJECT_GENERAL_SYMPTOM_LANG_1, _buf,
                                    &_leading, &_ascent, &_descent );

_from.v = _from2.v;
_from.h = _rct_chp_list.right - 3 - _width;
DisplayWindowObject ( win, &_from, &_to, OBJECT_GENERAL_SYMPTOM_LANG_1, _buf,
                      "", _attrib );
_from.h = _rct_chp_list.left  - 3;
_to.h   = _rct_chp_list.right + 3;

KeepViewBoenObject ( OBJECT_VBOEN_CHAPTER, &_from, &_to, _i,
                        VBOEN_KEEP_OBJECT, VBOEN_KEEP_NO_DISPLAY );


               /*********************************/
               /* Affichage du numero de symptomes */
               /*********************************/

        /* c1 = SptInMemory->Qualif;
         for(num =1; c1 !=0; c1>>=1) {
                                if (c1 & 1)
                                {
                                RectButton[num-1].Button = OFF;
                                DisplayVBoenButton(num-1);
                                 }
                                ++num ;
                                }
         c1 = SptInMemory->Degrees;
         for(num =1; c1 !=0; c1>>=1) {
                                if (c1 & 1)
                                {
                                RectButton[num+7].Button = OFF;
                                DisplayVBoenButton(num+7);
                                 }
                                ++num ;
                                }   */



                                        }
                                }         /* end of :   if ( _st ) */

                                if (_lang[1] == -1) _j++;
                        } /* end du FOR pour le 2eme. texte du symptome */

                        _from.v = _next_verti + VBOEN_SYMPTOM_INTERLIGNE;
                        _from.h = _rct_spt_list.left;
                        if (_st) _i++;

                        if ( _i >= (int)SC[WinViewBoen->SCNumber].NbSpt )
                                _st = FALSE;
                        else
                                SptInMemory = SptInMemory->Next;
                }

                WinViewBoen->LastSpt = _i;

                if ( mode != VBOEN_REFRESH ) {
                        if ( WinViewBoen->FirstSpt == 0 ) {
                        SetWindowScrollBar ( xvt_win_get_ctl ( WinViewBoen->Win, W_VBOEN_VSCROLL ),
                                                HVSCROLL, 1L, (long)SC[WinViewBoen->SCNumber].NbSpt,
```

```
                                        (long) 1 );
                }
                else {
                SetWindowScrollBar ( xvt_win_get_ctl ( WinViewBoen->Win, W_VBOEN_VSCROLL ),
                                HVSCROLL, 1L, (long)SC[WinViewBoen->SCNumber].NbSpt,
                                (long)( WinViewBoen->FirstSpt ) );
                }
        }

        SetViewBoenCurrentSymptom ( WinVi ewBoen->Win );

        if ( WinViewBoen->FirstDisplay ) {
                WinViewBoen->FirstDisplay = FALSE;
                WinViewBoen->Timer = xvt_timer_create ( win, 50 );
        }
}
/* Affichage des nombres de Spts. dans chaque clipboards */

/* on doit effacer l'emplacement o- on va ,crire */
/*
! 7023
*/
xvt_vobj_get_outer_rect ( xvt_win_get_ctl ( win, W_VBOEN_SC_LOC ), &_rct1 );
_rct.top = _rct1.bottom + 1;
_rct.left = _rct1.left;
xvt_vobj_get_outer_rect ( xvt_win_get_ctl ( win, W_VBOEN_SC_CONC ), &_rct1 );
_rct.right = _rct1.right;
_rct.bottom = WinViewBoen->SymptomRect.top - 1;
WindowDrawRect ( win, &_rct, (COLOR)xvt_vobj_get_attr ( win, ATTR_BACK_COLOR ) );

xvt_R3FNT_select_font( FF_HELVETICA, FF_SYSTEM, (short)8, &Font_For_Spt_Number );
        /* { xvt_font_set_family( * &Font_For_Spt_Number , XVT_FFN_HELVETICA );
                xvt_font_set_style ( * &Font_For_Spt_Number , XVT_FFN_SYSTEM );
                xvt_font_set_size  ( * &Font_For_Spt_Number , (short)7 ); },
                (XVT_FNTID?!)* &Font_For_Spt_Number ;
        */
xvt_R3FNT_win_set_font( win, &F ont_For_Spt_Number, FALSE);
        /* if the expression: (FALSE) is TRUE, set the XVT_FS_SCALE by:
                XVT_FONT_STYLE_MASK                                     style                                   =
xvt_font_get_style((XVT_FNTID?!)*&Font_For_Spt_Number...);
                xvt_font_set_style((XVT_FNTID?!)*&Font_For_Spt_Number..., style | XVT_FS_SCALE
);
                then set font:
                xvt_dwin_set_font(win, (XVT_FNTID?!)*&Font_For_Spt_Number...);
        */
xvt_dwin_get_font_metrics ( win, &_leading2, &_ascent2, &_descent2 );

for ( _j = 0; _j <4; _j++ ) {

        sprintf(_buf, "%d", SC[_j].NbSpt);

        xvt_vobj_get_outer_rect ( xvt_win_get_ctl ( win, W_VBOEN_SC_LOC + _j ), &_rct1 );
        _width = xvt_dwin_get_text_width ( win, _buf, strlen ( _buf ) );

        _from.v = _rct1.bottom + _leading2 + _ascent2 + _descent2 + 2;
        _from.h = _rct1.left + ( _rct1.right - _rct1.left - _width ) / 2;
        xvt_rect_set( &_rct, _from.h - 1, _rct1.bottom + 1, _from.h + _width + 1, _from.v + 1 );
        if (WinViewBoen->SCNumber == _j) {
                xvt_dwin_set_back_color ( win, COLOR_BLACK );
                xvt_dwin_set_fore_color ( win, Option.Repertory.ColorScreen );
                WindowDrawRect ( win, &_rct, COLOR_BLACK );
        }
        else {
                xvt_dwin_set_fore_color ( win, COLOR_BLACK );
                xvt_dwin_set_back_color ( win, Option.Repertory.ColorScreen );
                WindowDrawRect ( win, &_rct, Option.Repertory.ColorScreen );
        }
        xvt_dwin_draw_text ( win, _from.h, _from.v - _descent2, _buf, strlen ( _buf ) );
}

/*****************************************************************************/
/* Affichage des titres des colonnes des remedes, numero de symptomes et de qualifs. */
/*****************************************************************************/
xvt_dwin_set_fore_color ( win, COLOR_BLACK );
xvt_dwin_set_back_color ( win, Option.Repertory.ColorScreen );

xvt_rect_set ( &_rct, WinViewBoen->ChapterRect.left - 5,
                WinViewBoen->ChapterRect.top - _leading2 - _ascent2,
                WinViewBoen->ChapterRect.right - 5,
                WinViewBoen->ChapterRect.top );
WindowDrawRect ( win, &_rct, Option.Repertory.ColorScreen );

strcpy (_buf,  GetMessage (FileMessage, 4027, Option.Genera l.Lang, &RecMessage) );
xvt_dwin_draw_text ( win, _rct.left + 4, _rct.top + 4, _buf, strlen ( _buf ) );
```

```
        xvt_rect_set ( &_rct, WinViewBoen->NbRemediesRect.left - 5,
                       WinViewBoen->NbRemediesRect.top - _leading2 - _ascent2,
                       WinViewBoen->NbRemediesRect.right - 5,
                       WinViewBoen->NbRemediesRect.top );
        WindowDrawRect ( win, &_rct, Option.Repertory.ColorScreen );

        strcpy (_buf,  GetMessage (FileMessage, 800, Option.General.Lang, &RecMessage) );
        xvt_dwin_draw_text ( win, _rct.left + 4, _rct.top + 4, _bu f, strlen ( _buf ) );




        /************************************/
        /* On doit remettre le font courant */
        /************************************/

        xvt_R3FNT_win_set_font( win,
                                &Option.Object [ OBJECT_GENERAL_SYMPTOM_LANG_1 ].Font,
                                FALSE );

        xvt_win_set_cursor ( TASK_WIN, CURSOR_ARROW );
}

/*****************************************************************************

  NAME
                KeepViewBoenSelect - Keep the position of a symptom selected object

  SYNTAX
                void
                KeepViewBoenSelect ( type, from, to, id, shift_clic )

  PARAMETERS
                int             type;       I   Object type;
                PNT             *from,      I   Start point for the object
                                 *to;        I   End point for the object
                unsigned long   id;         I   Id of the object
                int             shift_clic; I   TRUE if the selection comes from a
                                                shift + clic, FALSE else
  INCLUDE
                xvt.h
                radar.h

  DESCRIPTION
                This function add an object to the selected object list for a taken
                symptoms window.

                If 'shift_clic' is TRUE, and if the object is already selected,
                this object is deleted from the selected list.

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************************/

void
KeepViewBoenSelect ( type, from, to, id, shift_clic )
        int             type;
        PNT             *from,
                        *to;
        unsigned long   id;
        int             shift_clic;
{
VBOEN_OBJECT        *_vobj,  /* *_tmp, */
                    *_next;
int                    _found;

        /*from-=3;*/

        if ( !shift_clic ) {


                KeepViewBoenObject ( type, from, to, id,
                                     VBOEN_KEEP_SELECT, VBOEN_KEEP_DISPLAY );



        }
        else {
```

```
                _found = FALSE;
                _vobj = _next = WinViewBoen->Select;
                while ( _next && !_found ) {
                        if ( _next->Type == (unsigned short)type && _next->Id == id ) {
                                _found = TRUE;
                        }
                        else {
                                _vobj = _next;
                                _next = _next->Next;
                        }
                }



                if ( !_found ) {


                        KeepViewBoenObject ( type, from, to, id,
                                             VBOEN_KEEP_SELECT, VBOEN_KEEP_DISPLAY );



                }
                else {


                        DisplayViewBoenSelect ( WinViewBoen->Win, _next, ATTRIBUTE_NORMAL );


                          /* _vobj */
                        if ( _next == WinViewBoen->Select )
                                WinViewBoen->Select = _next->Next;
                        else
                                _vobj->Next = _next->Next;

                        Free ( (char *)_next );
                }
        }

}

/*****************************************************************************

  NAME
                KeepViewBoenObject - Keep the position of a symptom window object

  SYNTAX
                void
                KeepViewBoenObject ( type, from, to, id, keep, display )

  PARAMETERS
                PNT             *from,      I   Start point for the object
                                                        *to;        I   End point for the object
                unsigned long   id;         I   Id of the object
                int             keep,       I   Type of keep, object or selected objet
                                                    VBOEN_KEEP_OBJECT or
                                                    VBOEN_KEEP_SELECT
                                display;    I   After keep, have to display object ?
                                                VBOEN_KEEP_DISPLAY or
                                                VBOEN_KEEP_NO_DISPLAY.

  INCLUDE
                xvt.h
                radar.h

  DESCRIPTION
                This function add the position of an obje ct to the object list of
                a ViewBoen window.

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************************/

void
KeepViewBoenObject ( type, from, to, id, keep, display )
        int             type;
```

```
        PNT             *from,
                                                *to;
        unsigned long   id;
        int             keep,
                                                display;
{
VBOEN_OBJECT        *_new;

        if ( _new = (VBOEN_OBJECT *) Malloc ( sizeof ( VBOEN_OBJECT ) ) ) {
                _new->Type = type;
                xvt_rect_set ( &_new->Rect, from->h, from->v, to->h, to->v );

                _new->Id = id;

                if ( keep == VBOEN_KEEP_OBJECT ) {
                        _new->Next = WinViewBoen->Object;
                        WinViewBoen->Object = _new;
                }
                else {
                        _new->Next = WinViewBoen->Select;
                        WinViewBoen->Select = _new;
                        if ( display == VBOEN_KEEP_DISPLAY ) {
                                DisplayViewBoenSelect ( WinViewBoen->Win, _new, ATTRIBUTE_REVERSE );
                        }
                }
        }
        else
                xvt_dm_post_error( GetMessage ( FileMessage, 17, Option.General.Lang,
                                                &RecMessage ) );
}


/*****************************************************************************

  NAME
                DisplayViewBoenSelect - Display a selected object

  SYNTAX
                void
                DisplayViewBoenSelect ( win, object, attrib )

  PARAMETERS
                WINDOW              win;          I   Window for display
                VBOEN_OBJECT     type;         I   Type of object to display
                int                 attrib;       I   Attribute for display
                                        ATTRIBUTE_NORMAL or ATTRIBUTE_REVERSE

  INCLUDE
                xvt.h
                radar.h

  DESCRIPTION
                Display a selected object in a ViewBoen window.

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************************/

void
DisplayViewBoenSelect ( win, object, attrib )
        WINDOW              win;
        VBOEN_OBJECT     *object;
        int                 attrib;
{
SYMPTOM_IN_MEMORY  *SptInMemory;
char               _buf [20];

short              _leading, _descent, _ascent;

int                _i,
                   _width,
                   _lang    [ 2 ],
                   _typetxt [ 2 ],
                   _typetext,
                   _size_for_spt_nb;

unsigned char      *_text [ 2 ];

PNT                _from,
```

```
                _to;


        SptInMemory = SC[WinViewBoen->SCNumber].First;
        for (_i=0; _i < (int)object->Id; _i++) {
                SptInMemory = SptInMemory->Next;
        }


        if ( object->Type == OBJECT_VBOEN_SYMPTOM ) {
                _lang [ 0 ] = _lang [ 1 ] = -1;
/*
                if ( WinViewBoen->Rep->Repertory->Language & REPERTORY_LANG_1 &&
                        WinViewBoen->Rep->Repertory->Lang1 ) {
*/
                        _text    [ 0 ] = SptInMemory->SptTextLg1;
                        if ( SptInMemory->Langue [0] == LANG_UNDEFINED ) {
                                _lang [0] = -1;
                        }
                        else {
                                _lang [ 0 ] = SptInMemory->Langue[0];
                        }

                        _typetxt [ 0 ] = OBJECT_GENERAL_SYMPTOM_LANG_1;


                        _text    [ 1 ] = SptInMemory->SptTextLg2;

                        _typetxt [ 1 ] = OBJECT_GENERAL_SYMPTOM_LANG_2;
                _from.v = object->Rect.top;
                _from.h = object->Rect.left;



                /* Calculates needed space for the symptom number */
                if ( _lang[0] == -1 ) _typetext = _typetxt[1];
                else                  _typetext = _typetxt[0];

                sprintf(_buf, "%d. ", _i+1);
                _size_for_spt_nb = GetWindowObjectDim ( win, _typetext, _buf,
                                                &_leading, &_ascent, &_descent );


                /* Displays the symptom number */


                DisplayRepertorySentense ( win, _typetext, _buf, "",
                                        FALSE, &_from, &_to, attrib | ATTRIBUTE_CLEAR,
                                        object->Rect.left,
                                        WinViewBoen->SymptomRect.right - 2,
                                        REPERTORY_CANNOT_SPLIT );



                _from.h += _size_for_spt_nb;


                /* Displays First Language */
                if ( _lang[0] != -1) {
                        DisplayRepertorySentense ( win, _typetxt[0], _text[0], "",
                                        FALSE, &_from, &_to, attrib | ATTRIBUTE_CLEAR,
                                        object->Rect.left,
                                        WinViewBoen->SymptomRect.right - 2,
                                        REPERTORY_CANNOT_SPLIT );
                        if (_lang[1] != -1 ) _from.v = _to.v;
                }

                if ( _lang[1] != -1 ){
                        _from.h =   object->Rect.left;
                        _from.h +=  _size_for_spt_nb;
                        if ( _lang[0] != -1) {
                                _from.h +=  VBOEN_OFFSET_FOR_SECOND_LANG;
                        }

                        DisplayRepertorySentense ( win, _typetxt[1], _text[1], "",
                                                FALSE, &_from, &_to, attrib | ATTRIBUTE_CLEAR,
                                                object->Rect.left,
                                                WinViewBoen->SymptomRect.right - 2,
                                                REPERTORY_CANNOT_SPLIT );
                }

                return;
        }
```

```
          /* REMEDY */

        if ( object->Type == OBJECT_VBOEN_REMEDY) {
                sprintf(_buf, "%d", NumberOfDifferentRemedies (SptInMemory) );

                _width = GetWindowObjectDim ( win, OBJECT_GENERAL_SYMPTOM_LANG_1, _buf,
                                              &_leading, &_ascent, &_descent );

                _from.h = object->Rect.right - 4 - _width;
                _from.v = object->Rect.top;
                DisplayWindowObject ( win, &_from, &_to, OBJECT_GENERAL_SYMPTOM_LANG_1, _buf,
                                      "", attrib | ATTRIBUTE_CLEAR );

                return;
        }


}
```

```
/*****************************************************************************

  NAME
                ViewBoenKeyboardHandler - Manage the keyboard keys in the ViewBoen
                window .

  SYNTAX
                void
                ViewBoenKeyboardHandler ( win, event )

  PARAMETERS
                WINDOW          win;        I   ViewBoen window
                EVENT           *event;     I   Event received

  INCLUDE
                xvt.h
                radar.h

  DESCRIPTION
                Manage the keyboard keys in the ViewBoen window

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************************/

void
ViewBoenKeyboardHandler ( win, event )
        WINDOW      win;
        EVENT       *event;
{
int               _i,
                  _j;
VBOEN_OBJECT    _ptr,
                *_object;

PNT               _to,
                  _from;


        switch ( event->v.chr.ch ) {

                case K_F1:
                        RadarHelp ( win, 3 );
                        break;

                case '1' :
                case K_KP1:
                        xvt_dm_post_note("control = %d",event->v.chr.control);
                        xvt_dm_post_note("shift = %d",event->v.chr.shift);
                        if ( event->v.chr.control ){
                        xvt_dm_post_note("ok");
                        ManageCheckKeyboardButton ( (int) 1);}
```

```
        else
        ManageVBoenKeyboardButton ( (int)1);
        break;

case '2' :
        if ( event->v.chr.control ){
        xvt_dm_post_note("ok");
        ManageCheckKeyboardButton ( (int) 2);}
        else
        ManageVBoenKeyboardButton ( (int)2);
        break;

case '3' :
        if ( event->v.chr.control ){
        xvt_dm_post_note("ok");
        ManageCheckKeyboardButton ( (int) 3);}
        else
        ManageVBoenKeyboardButton ( (int)3);
        break;

case '\t':
case K_BTAB:
       WinViewBoen->Focus = WindowTabHandler ( win, event, WinViewBoen->Focus,
                        W_VBOEN_OK, W_VBOEN_VSCROLL, TRUE, W_VBOEN_OK );
       break;

case 27:
       TerminateViewBoenWindow ( win );
       break;


case 13:
case 10:
       switch ( WinViewBoen->Focus ) {
              case W_VBOEN_OK :
                     TerminateViewBoenWindow ( WinViewBoen->Win );
                     break;

              case W_VBOEN_HELP :
                     RadarHelp ( win, (int) 3 );
                     break;

              case W_VBOEN_CHECK :


                     break;


              case W_VBOEN_ANALYS :
         if (GoToAnalyse())
            {
             WinViewBoen->StartAnalysis = TRUE;
             TerminateViewBoenWindow ( win );
             }
       else xvt_dm_post_note(" Nombre de symptoms complets insuffisant");

                     break;

              case W_VBOEN_SC_LOC :
                     ChangeViewBoenClipboard ( win, (int) 0);
                     break;

              case W_VBOEN_SC_SENS :
                     ChangeViewBoenClipboard ( win, (int) 1);
                     break;

              case W_VBOEN_SC_MOD :
                     ChangeViewBoenClipboard ( win, (int) 2);
                     break;

              case W_VBOEN_SC_CONC :
                     ChangeViewBoenClipboard ( win, (int) 3);
                     break;


              default:
                     if ( SC [WinViewBoen->SCNumber].NbSpt > 0 ) {
                            _i  = WinViewBoen->CurrentSpt;
                            SptInMemory = SC [WinViewBoen->SCNumber].First;
                            for (_j=0; _j < _i && SptInMemory->Next != NULL; _j++)
                                   SptInMemory = SptInMemory->Next;

                            InitializeModifSymptomValues  ( );
                     }
                     break;
```

```
                        }
                        break;

                case K_NEXT:
                        if ( WinViewBoen->LastSpt < (int)SC [WinViewBoen->SCNumber].NbSpt ) {
                                DisplayViewBoenWindow ( win, VBOEN_PAGE_DOWN );
                        }
                        break;

                case K_PREV:
                        if ( WinViewBoen->FirstSpt > 0 ) {
                                DisplayViewBoenWindow ( win, VBOEN_PAGE_UP );
                        }
                        break;

                case K_DOWN:
                        if ( WinViewBoen->CurrentSpt < (int)SC [WinViewBoen->SCNumber].NbSpt ) {
                                ChangeViewBoenLine ( win, VBOEN_LINE_DOWN );
                        }
                        break;

                case K_UP:
                        if (WinViewBoen->CurrentSpt != 0) {
                                ChangeViewBoenLine ( win, VBOEN_LINE_UP );
                        }
                        break;

                case K_HOME:


        case K_LHOME:
           if ( event->v.chr.control ) {
              xvt_dm_post_note("ok");
              DisplayViewBoenWindow ( win, VBOEN_GO_TO_FIRST );
                        }
                        break;

        case K_END:
           if ( event->v.chr.control ) {
              xvt_dm_post_note("ok");
              DisplayViewBoenWindow ( win, VBOEN_GO_TO_LAST );
           }
           break;


        case K_DEL:
        case '\b':
           BDeleteSymptoms ( );
           break;

        case ' ' :

           _ptr.Type = OBJECT_VBOEN_SYMPTOM;
           _ptr.Id   = (long) WinViewBoen->CurrentSpt;

           _object = (VBOEN_OBJECT *) NULL;
           _object = GetViewBoenObjectInList ( WinViewBoen->Object, &_ptr );

           if ( _object ) {
                                if ( !event->v.chr.control ) {
                 FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_DISPLAY );
              }
                                _from.v = _object->Rect.top;
              _from.h = _object->Rect.left;
              _to.v   = _object->Rect.bottom;
              _to.h   = _object->Rect.right;
              KeepViewBoenSelect ( _object->Type,
                                   &_from,
                                   &_to,
                                   _object->Id,
                                   event->v.chr.control );

           }
           break;

                default:
           break;
     }
}


/*****************************************************************************
```

```
  NAME
       ManageViewBoenUserEvent - Manage the user event

  SYNTAX
       void
       ManageViewBoenUserEvent ( win, event )

  PARAMETERS
       WINDOW          win;        I    Window reveiving the event
       EVENT           *event;     I    Event received

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
                This function manage the user event for this window.

                Called when user type the F1 key or an acceleration key.

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************************/

void
ManageViewBoenUserEvent ( win, event )
   WINDOW      win;
   EVENT       *event;
{
 switch ( event->v.user.id ) {

    case RADAR_ACCEL_KEY:
               switch ( GetControlIdForAccelKey ( (WIN_MSG *) &WinViewBoenMsg,
                          (unsigned char) event->v.user.ptr ) ) {

                     case W_VBOEN_OK :
                            TerminateViewBoenWindow ( WinViewBoen->Win );
                            break;

                            case W_VBOEN_HELP :
                                   RadarHelp ( win, 3 );
                                   break;

                            case W_VBOEN_CHECK :
                                   break;


                            case W_VBOEN_ANALYS :
                             if (GoToAnalyse()){
                                 WinViewBoen->StartAnalysis = TRUE;
                                TerminateViewBoenWindow ( win );
                             }
                            else xvt_dm_post_note(" Nombre de symptoms complets insuffisant");
                            break;

                            case W_VBOEN_SC_LOC :
                                   ChangeViewBoenClipboard ( win, (int) 0);
                                   break;

                            case W_VBOEN_SC_SENS :
                                   ChangeViewBoenClipboard ( win, (int) 1);
                                   break;

                            case W_VBOEN_SC_MOD :
                                   ChangeViewBoenClipboard ( win, (int) 2);
                                   break;

                            case W_VBOEN_SC_CONC :
                                   ChangeViewBoenClipboard ( win, (int) 3);
                                   break;


                                          default:
                                          xvt_scr_beep ();
                                          break;
                                   }
                                   break;

        case -1:
        default:
```

```
        break;
    }
}



/*****************************************************************************

  NAME
                GetViewBoenObjectAtPoint - Search for an object at a point

  SYNTAX
                VBOEN_OBJECT
                *GetViewBoenObjectAtPoint ( point, keep )

  PARAMETERS
                PNT             *point;  I   Point for the object
                int             keep;    I   Type of object kept
                                             VBOEN_KEEP_OBJECT or
                                             VBOEN_KEEP_SELECT

  INCLUDE
                xvt.h
                radar.h

  DESCRIPTION
                This function search in a specific ViewBoen window in the list of
                displayed object or selected objects for an object being at a
                specific point.

  RETURN
                Pointer to the found object or NULL if no object at this point.

  EXAMPLE

  SEE ALSO

*****************************************************************************/

VBOEN_OBJECT
*GetViewBoenObjectAtPoint ( point, keep )
        PNT                 *point;
        int                 keep;
{
int                 _st;
VBOEN_OBJECT     *_ptr;

        _st = FALSE;


        if ( keep == VBOEN_KEEP_OBJECT )
                _ptr = WinViewBoen->Object;
        else
                _ptr = WinViewBoen->Select;


   while ( !_st && _ptr ) {


      _st = xvt_rect_has_point ( &_ptr->Rect, *point );
      if ( !_st )
        _ptr = _ptr->Next;
   }


   return ( _st ? _ptr : (VBOEN_OBJECT *) NULL );

}



/*****************************************************************************

  NAME
      ViewBoenMouseHandler - Handle mouse event on a ViewBoenwindow

  SYNTAX
      void
      ViewBoenMouseHandler ( win, event )
```

```
  PARAMETERS
       WINDOW     win;            I   Window receiving the event
       EVENT      *event;         I   Event received

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
       This function manages all mouse event in a remedy window.

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************************/

void
ViewBoenMouseHandler ( win, event )
   WINDOW   win;
   EVENT    *event;
{
PNT                          _from,
                            _to;
int                          _new_selected,
                            _i,
                            _j;
unsigned int                 _nb_object;

static VBOEN_OBJECT         *_object,
                             *_ptr,
                             *_select;
static int                   _drawing = FALSE;
static RCT                   _rect;
static BOOLEAN               _first;
static PNT                   _point;


   _new_selected = FALSE;
   switch ( event->type ) {

      case E_MOUSE_DOWN :
         _object = GetViewBoenObjectAtPoint ( &eve nt->v.mouse.where,
                           VBOEN_KEEP_OBJECT );


         _drawing = TRUE;
         _first = TRUE;
         _rect.left = event->v.mouse.where.h;
         _rect.top = event->v.mouse.where.v;
         _rect.right = event->v.mouse.where.h;
         _rect.bottom = event->v.mouse.where.v;
         _point = event->v.mouse.where;
         xvt_win_trap_pointer ( win );


                     break;

      case E_MOUSE_MOVE :

         if ( !_drawing )              return;
         if ( _point.h == event->v.mouse.where.h &&
              _point.v == event->v.mouse.where.v     )   return;

         if ( !_first )  WindowDrawRubberRect ( win, &_rect );
         _first = FALSE;
         _rect.right = event->v.mouse.where.h;
         _rect.bottom = event->v.mouse.where.v;
         _point = event->v.mouse.where;
         WindowDrawRubberRect ( win, &_rect );


         break;


      case E_MOUSE_UP :


         if ( _object && _object == GetViewBoenObjectAtPoint ( &event->v.mouse.where,
                VBOEN_KEEP_OBJECT ) ) {

            if ( _drawing ) {
```

```
      WindowNormalizeRect ( &_rect, &_rect );
      WindowDrawRubberRect ( win, &_rect );
   }


   if ( _object->Type != OBJECT_VBOEN_REMEDY ) {
                  /* Select the item where clicked */

      if ( !event->v.mouse.control ) {

         FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_DISPLAY );

      }

      _from.v = _object->Rect.top;
      _from.h = _object->Rect.left;
      _to.v = _object->Rect.bottom;
      _to.h = _object->Rect.right;
      KeepViewBoenSelect ( _object->Type,
                           &_from,
                           &_to,
                           _object->Id,
                           event->v.mouse.control );


   }

   _drawing = FALSE;

   if ( _object->Type == OBJECT_VBOEN_SYMPTOM ||
        _object->Type == OBJECT_VBOEN_REMEDY  ) {

      WinViewBoen->CurrentSpt = (int)_object->Id;
      SetViewBoenCurrentSymptom ( win );
   }
}


xvt_win_release_pointer ();
if ( _drawing ) {


   WindowNormalizeRect ( &_rect, &_rect );
   if ( !_first )   WindowDrawRubberRect ( win, &_rect );

                    /* Search for objects in draw rect */

   /*********************************************************************/
   /* On doit copter le nombre d'object affic,s car on doit consulter */
   /* la liste des objects affiches dans le sens inverse de maniŠre   */
   /* … ce que quand on fait un eventuel PASTE des sympt"mes          */
   /* s,lectionn,s, ils soient copi,s dans l'ordre o— ils se trouvent */
   /*********************************************************************/
   _ptr = WinViewBoen->Object;
   _nb_object = 0;
   while ( _ptr ) {
      _ptr = _ptr->Next;
      _nb_object ++;
   }

   _i=_nb_object-1;
   _ptr = WinViewBoen->Object;
                    while ( _i >= 0 ) {

      _ptr = WinViewBoen->Object;
      for (_j=0; _j<_i; _j++) _ptr = _ptr->Next;

      if ( xvt_rect_intersect ( NULL, &_ptr->Rect, &_rect ) ) {
         if ( _ptr->Type != OBJECT_VBOEN_REMEDY ) {
            if ( (!event->v.mouse.control) && _drawing ) {
               FreeViewBoenObject ( VBOEN_KEEP_SELECT,
                                    VBOEN_KEEP_DISPLAY );

            }

            _from.v = _ptr->Rect.top;
            _from.h = _ptr->Rect.left;
            _to.v   = _ptr->Rect.bottom;
            _to.h   = _ptr->Rect.right;

            KeepViewBoenSelect ( _ptr->Type,
                                 &_from,
                                 &_to,
                                 _ptr->Id,
                                 event->v.mouse.control );
```

```
                    _new_selected = TRUE;
                    _drawing = FALSE;
                 }
              }
              _i --;
           }
/*
           _ptr = WinViewBoen->Object;
           while ( _ptr ) {
              if ( xvt_rect_intersect ( NULL, &_ptr->Rect, &_rect ) ) {
                 if ( _ptr->Type != OBJECT_VBOEN_REMEDY ) {
                    if ( (!event->v.mouse.control) && _drawing ) {
                       FreeViewBoenObject ( VBOEN_KEEP_SELECT,
                                            VBOEN_KEEP_DISPLAY );
                    }

                    _from.v = _ptr->Rect.top;
                    _from.h = _ptr->Rect.left;
                    _to.v   = _ptr->Rect.bottom;
                    _to.h   = _ptr->Rect.right;

                    KeepViewBoenSelect ( _ptr->Type,
                                         &_from,
                                         &_to,
                                         _ptr->Id,
                                         event->v.mouse.control );
                    _new_selected = TRUE;
                    _drawing = FALSE;
                 }
              }
              _ptr = _ptr->Next;
           }
*/

           if ( WinViewBoen->Select && _new_selected ) {
              WinViewBoen->CurrentSpt = (int)WinViewBoen->Select- >Id;
              SetViewBoenCurrentSymptom ( win );
           }
        }
        ManageVBoenButton ( &event->v.mouse.where, event->v.mouse.control);
        ManageCheckButton ( &event->v.mouse.where, event->v.mouse.control);
        _object = NULL;
        _drawing = FALSE;
        break;

     case E_MOUSE_DBL :
        _object = GetViewBoenObjectAtPoint ( &event->v.mouse.where,
                          VBOEN_KEEP_OBJECT );
        /*xvt_dm_post_note ("Double clic on object type %d, Id = %ld", _object->Type, _object->Id );*/

        /*   if ( xvt_rect_has_point ( &WinViewBoen ->NbRemediesRect, event->v.mouse.where ) )
             xvt_dm_post_note ( "Double clic on the NbRemediesRect" );*/



        if ( _object ) {
           if ( _object->Type == OBJECT_VBOEN_SYMPTOM  ||
                _object->Type == OBJECT_VBOEN_REMEDY )  {
              /* On se positionne ce Sympt"me en m,moire */
              _i  = (int)_object->Id;


              SptInMemory = SC [WinViewBoen->SCNumber].First;
              for (_j=0; _j < _i && SptInMemory->Next != NULL; _j++)
                 SptInMemory = SptInMemory->Next;

              /*if ( _object->Type == OBJECT_VBOEN_SYMPTOM ) {
                 InitializeModifSymptomValues  ( );
              } */

              if ( _object->Type == OBJECT_VBOEN_REMEDY )  {
                 InitializeRemedyWindow ( WinRep, 0L, 0 );
              }
                              }
        }

        else {

          /* if ( xvt_rect_has_point ( &WinViewBoen->CompSympRect, event->v.mouse.where ) )
              xvt_dm_post_note ( "Double clic on the Completesymptom" );
            */
        }

        break;
```

```
    }
}



/*****************************************************************************

  NAME
       ChangeViewBoenClipboard- Change the displayed clipboard

  SYNTAX
       void
       ChangeViewBoenClipboard ( new_sc)

  PARAMETERS
       int                 new_sc; Number of the new clipboard to display.

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
       Change the displayed clipboard

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************************/
void
ChangeViewBoenClipboard ( win, new_sc )
int       new_sc;
WINDOW    win;
{

   /* Frees the current lists of objects */
   FreeViewBoenObject ( VBOEN_KEEP_OBJECT, VBOEN_KEEP_NO_DISPLAY );
   FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_NO_DISPLAY );

   WinViewBoen->SCNumber = new_sc;

#if XVTWS != MACWS
   xvt_dwin_clear (win, (COLOR)xvt_vobj_get_attr (win, ATTR_BACK_COLOR) );
#endif
   DisplayViewBoenWindow ( win, VBOEN_REFRESH );
   xvt_vobj_raise( win );

}

/*****************************************************************************

  NAME
       SetViewBoenCurrentSymptom ( win );

  SYNTAX
       void
       SetViewBoenCurrentSymptom ( win );

  PARAMETERS
       WINDOW    win;


  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
       Displays the finger in front of the current symptom

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************************/

void
SetViewBoenCurrentSymptom ( win )
WINDOW           win;
{
```

```
RCT                 _rect_finger, _rcti;
int                 _j, _k,num;
VBOEN_OBJECT     *_vobj, *_ptr;


   /* Clear the previous Current Symptom */
   xvt_rect_set (&_rect_finger, WinViewBoen->SymptomRect.left + 1,
                              WinViewBoen->SymptomRect.top + 1,
                              WinViewBoen->SymptomRect.left + 21,
                              WinViewBoen->SymptomRect.bottom - 2 );
   WindowDrawRect (win, &_rect_finger, (COLOR) xvt_vobj_get_attr (win, ATTR_BACK_COLOR) );


        /* Identify the symptom to be the current symptom */
   _j = WinViewBoen->LastSpt - WinViewBoen->CurrentSpt;

   _vobj = WinViewBoen->Object;
   while (_vobj->Type != OBJECT_VBOEN_SYMPTOM) _vobj = _vobj->Next;

   _k = 0;
   while (_k < _j && _vobj) {
      if (_vobj->Type == OBJECT_VBOEN_SYMPTOM) _k++;
      if (_k < _j) _vobj = _vobj->Next;
   }


   /* Draws the Finger */
   xvt_rect_set (&_rect_finger, _vobj->Rect.left-22,
                           _vobj->Rect.top+3,
                           _vobj->Rect.left-4,
                           _vobj->Rect.top+13);

   xvt_rect_set ( &_rcti, 0, 0, 0, 0 );
   xvt_image_get_dimensions ( PictureHand, &_rcti.right, &_rcti.bottom );
   xvt_dwin_draw_image ( win, PictureHand, &_rect_finger, &_rcti );

     for (num = 0; num <16; num++)
       {
        RectButton[num].Button = ON;
        DisplayVBoenButton ( num);
        }

       if(WinViewBoen->Select)
          { _ptr = WinViewBoen->Select;
             while ((_ptr) && (_ptr->Type == OBJECT_VBOEN_SYMPTOM))
                    {
                     RefreshButton(WinViewBoen->SCNumber, (int)_ptr->Id);
                     _ptr = _ptr->Next;
                     }
           }
         else
     /* RefreshSelectSymptom ();*/
      RefreshButton(WinViewBoen->SCNumber, WinViewBoen->CurrentSpt);
}


/*****************************************************************************

  NAME
       ChangeViewBoenLine - Moves the Current Symptom

  SYNTAX
       int
       ChangeViewBoenLine ( win, mode )

  PARAMETERS
       WINDOW              win;        I   ViewBoen window
       int                 mode;       I   Mode for display
                                         VBOEN_LINE_UP
                                         VBOEN_LINE_DOWN

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION

  RETURN

  EXAMPLE

  SEE ALSO

*****************************************************************************/
```

```
void
ChangeViewBoenLine ( win, mode )
   WINDOW     win;
   int        mode;
{

   if ( mode == VBOEN_LINE_DOWN) {
      /* if current is last of the window --> Next Page */
      if ( WinViewBoen->CurrentSpt == (int)WinViewBoen->Object->Id ) {
         if (WinViewBoen->CurrentSpt+1 < (int)SC [WinViewBoen->SCNumber].NbSpt) {
            DisplayViewBoenWindow ( win, VBOEN_PAGE_DOWN );
                     }
      }
      else {
         if (WinViewBoen->CurrentSpt+1 < (int)SC [WinViewBoen->SCNumber].NbSpt) {
            WinViewBoen->CurrentSpt ++;
            SetViewBoenCurrentSymptom ( win );
         }
      }
   }
   else {
      /* if current is first of the page --> Previous Page */
      if ( WinViewBoen->CurrentSpt == (int)WinViewBoen->FirstSpt ) {
         if (WinViewBoen->CurrentSpt > 0) {
            DisplayViewBoenWindow ( win, VBOEN_LINE_UP );
         }
      }
      else {
         if (WinViewBoen->CurrentSpt > 0) {
            WinViewBoen->CurrentSpt --;
            SetViewBoenCurrentSymptom ( win );
         }
      }
       }
}


/*****************************************************************************/

  NAME
       CopieSptToSC - copy Symptom

  SYNTAX
       int
 CopySptToSC (_SCNumber,_type, _spt )

  PARAMETERS
   int                  _SCNumber
                        _type
   SYMPTOM_IN_MEMORY    _spt

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
            Copie spt (un symptôme) dans un clipboard spécifié par ( SCNumber)
  RETURN

  EXAMPLE

  SEE ALSO

/*****************************************************************************/
int
CopySptToSC (_SCNumber,_type, _spt )
int _SCNumber,_type;
SYMPTOM_IN_MEMORY     *_spt;
{
  int                   _i,
                        _j;

            /*********************************************************/
            /* Allocating a structure to take this symptom in memory */
            /*********************************************************/
            if ( !(SptInMemory =
               (SYMPTOM_IN_MEMORY *) Malloc (sizeof (SYMPTOM_IN_MEMORY)) ) ) {
               xvt_dm_post_note( GetMessage ( FileMessage, 84, Option.General.Lang, &RecMessage) );
               return ( RADAR_NOT_OK );
            }
            SptInMemory->Next = (SYMPTOM_IN_MEMORY *)NULL;
            SptInMemory->Check = DB_CHECK;
            /*********************************************/
            /* Allocation of memory for the symptom texts */
```

```
            /*********************************************/

            if ( !(SptInMemory->SptTextLg1 =
               (unsigned char *) Malloc ( strlen (_spt->SptTextLg1) + 2 )) )   {
               xvt_dm_post_note( GetMessage ( FileMessage, 88, Option.General.Lang, &RecMessage) );
               Free ( (unsigned char *) SptInMemory);
               return (RADAR_NOT_OK);
            }

            if ( !(SptInMemory->SptTextLg2 =
               (unsigned char *) Malloc ( strlen (_spt->SptTextLg2) + 2)) )   {
               xvt_dm_post_note( GetMessage ( FileMessage, 89, Option.General.Lang, &RecMessage) );
               Free ( (unsigned char *) SptInMemory->SptTextLg1);
               Free ( (unsigned char *) SptInMemory);
               return (RADAR_NOT_OK);
            }

            memset (SptInMemory->SptTextLg1, '\0', strlen (_spt->SptTextLg1));
            memset (SptInMemory->SptTextLg2, '\0', strlen (_spt->SptTextLg2));

            /*********************************************/
            /* On alloue de la memoire pour les remedes  */
            /*********************************************/

            _i =  (int) _spt->NbRemedies;
            _i += 2;

            if ( !(SptInMemory->RmdList =
                 (REMED_LIST *) Malloc ( (sizeof (REMED_LIST) * (_i)) ))  )   {
               xvt_dm_post_note( GetMessage ( FileMessage, 90, Option.General.Lang, &RecMessage) );
               Free ( (unsigned char *) SptInMemory->SptTextLg1);
               Free ( (unsigned char *) SptInMemory->SptTextL g2);
               Free ( (unsigned char *) SptInMemory);
               return (RADAR_NOT_OK);
            }

            /*********************************************/
            /* On recopie le symptome membres ... membres */
            /*********************************************/
            SptInMemory->Intensity  =  (char)_type     ;
            /*SptInMemory->Intensity  =  _spt->SCNumber   ;*/
            SptInMemory->Group      =  _spt->Group     ;
            SptInMemory->Degrees    =  _spt->Degrees   ;
            SptInMemory->Qualif     =  _spt->Qualif    ;
            SptInMemory->View       =  _spt->View      ;
            SptInMemory->NbRemedies =  _spt->NbRemedies;
            SptInMemory->Origin     =  _spt->Origin    ;
            SptInMemory->Langue[0]  =  _spt->Langue[0] ;
            SptInMemory->Langue[1]  =  _spt->Langue[1] ;

            strcpy( SptInMemory->SptTextLg1, _spt->SptTextLg1 );
            strcpy( SptInMemory->SptTextLg2, _spt->SptTextLg2 );

            for (_j=0; _j<(int)_spt->NbRemedies; _j++) {
               SptInMemory->RmdList[_j].Check    = DB_CHECK;
               SptInMemory->RmdList[_j].RemedId  = (unsigned short) _spt->RmdList [_j].RemedId;
               SptInMemory->RmdList[_j].Degree   = (unsigned char)  _spt->RmdList [_j].Degree;
               SptInMemory->RmdList[_j].AuthorId = (unsigned short) _spt->RmdList [_j].AuthorId;
            }
            SptInMemory->Next = (SYMPTOM_IN_MEMORY *)NULL;

            UpdateSymptomClipboard ( _SCNumber+1, SptInMemory->NbRemedies );

   return ( RADAR_OK );
}


/*****************************************************************************/

  NAME
 DisplayVBoenButton - Display number button

  SYNTAX
       void
DisplayVBoenButton ( num )

  PARAMETERS
   int                 num

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
            Display number(num) button
```

```
  RETURN

  EXAMPLE

  SEE ALSO

/*****************************************************************************/


void DisplayVBoenButton ( num )
    int      num;
{
   XVT_IMAGE      _picture;
   RCT            _rcti;
   unsigned char  _name [ 255 ],
                  _file [ 255 ];
   sprintf ( _name, "cl%d_%d.bmp", num + 1, RectButton[num].Button );
   strcpy ( _file, GetFullFileName ( Option.General.ImageDir, _name ) );

   DbVirtualClose (2);
   if ( ( _picture = xvt_image_read ( _file ) ) ) {
        xvt_rect_set ( &_rcti, 0, 0, 0, 0 );
        xvt_image_get_dimensions ( _picture, &_rcti.right, &_rcti.bottom );

        RectButton[ num ].Rect.left  = 2;
        RectButton[ num ].Rect.top   = 50;
        if ( num % 2 )
           RectButton[ num ].Rect.left  += _rcti.right + 2;
        RectButton[ num ].Rect.top  += ( ( _rcti.bottom + 2 ) * ( num / 2 ) );
        RectButton[ num ].Rect.right  = RectButton[ num ].Rect.left  + _rcti.right;
        RectButton[ num ].Rect.bottom = RectButton[ num ].Rect.top   + _rcti.bottom;

        xvt_dwin_draw_image ( WinViewBoen->Win, _picture, &(RectButton[ num ].Rect), &_rcti );
        xvt_image_destroy ( _picture );
   }

}

/*****************************************************************************

  NAME
void ManageVBoenButton - manage view button

  SYNTAX
void ManageVBoenButton ( point,ctr_click )

  PARAMETERS

PNT     *point;
int     ctr_click;

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
            Manage view button
            If (Ctrl + Click (on a button)= OK) display
                             Button ON  if button = OFF
                             Button OFF if Button = ON
  RETURN

  EXAMPLE

  SEE ALSO


*****************************************************************************/
void ManageVBoenButton ( point,ctr_click )
    PNT     *point;
    int     ctr_click;
{
 int          _i, _tampon;

 for ( _i = 0; (_i < 16)&&(!ctr_click); _i++ ) {
    if ( xvt_rect_has_point ( &(RectButton[ _i ].Rect ), *point ) )
       {
       _tampon = RectButton[_i].Button;
       if (_tampon== OFF)
       DeleteCurrentNumberSymptom (_i+1);
       if (_tampon == ON)
       GetNumberSymptom(_i+1);
       break;
```

```
        }
 }
}


/*********************************************************/
NAME
    ManageVBoenKeyboardButton -

  SYNTAX
 void ManageVBoenKeyboardButton ( _i )

  PARAMETERS

int     _i ;

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
            If Button= ON delete number button
            If Button= OFF get number button
RETURN

  EXAMPLE

  SEE ALSO

/*********************************************************/

void ManageVBoenKeyboardButton ( _i )
    int     _i;
{
 int           _tampon;

        _tampon = RectButton[_i-1].Button;
        if (_tampon== OFF)
        DeleteCurrentNumberSymptom (_i);
        if (_tampon == ON)
        GetNumberSymptom(_i);
}


/*********************************************************/
NAME
 ManageCheckButton

  SYNTAX
 void ManageCheckButton

  PARAMETERS

PNT     *point;
int    ctr_click;

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
            Display complete symptom number if Ctrl + click on button
RETURN

  EXAMPLE

  SEE ALSO


/*********************************************************/

void ManageCheckButton ( point, ctr_click )
    PNT     *point;
    int    ctr_click;
{
 int          _i;

 for ( _i = 0; (_i < 16)&&(ctr_click); _i++ ) {
    if ( xvt_rect_has_point ( &(RectButton[ _i ].Rect ), *point ) && ctr_click)
        {
        DisplayCompleteSymptom(_i+1);
        ChangeViewBoenClipboard(WinViewBoen->Win,4);
        break;
        }
 }
```

```
}

/*********************************************/
NAME
RefreshButton

  SYNTAX
Void
RefreshButton ( _SCNumber, _SptNumber)

  PARAMETERS
Int  _SCNumber;
Int  _SptNumber;

INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
            Display all button number for a symptom
RETURN

  EXAMPLE

  SEE ALSO


/*********************************************/

void
RefreshButton ( _SCNumber, _SptNumber)
   int _SCNumber;
   int _SptNumber;

{
unsigned char c1;
SYMPTOM_IN_MEMORY  *_spt;


int                _i, num;

       if( SC[_SCNumber].First == NULL ) {
/*xvt_dm_post_note( GetMessage ( FileMessage, 109, Option.General.Lang, &RecMessage), _SptNumber );*/
              return;
   }

   _i   = 0;

   _spt = SC[_SCNumber].First;
   while ( (_i < _SptNumber) &&
          (_spt != (SYMPTOM_IN_MEMORY *)NULL) ) {
           _spt = _spt->Next;
           _i++;
   }
   if (_i==_SptNumber)
   {
        c1 = _spt->Qualif;
        for(num =1; c1 !=0; c1>>=1) {
                                if (c1 & 1)
                                {
                                RectButton[num-1].Button = OFF;
                                DisplayVBoenButton(num-1);
                                 }
                                ++num ;
                                }
        c1 = _spt->Degrees;
        for(num =1; c1 !=0; c1>>=1) {
                                if (c1 & 1)
                                {
                                RectButton[num+7].Button = OFF;
                                DisplayVBoenButton(num+7);
                                 }
                                ++num ;
                                }

    }


}

/*********************************************************************/
```

```
NAME
RefreshSelectSymptom

  SYNTAX
Void
RefreshSelectSymptom ( )

  PARAMETERS

INCLUDE
      xvt.h
      radar.h

  DESCRIPTION
          Display all button number for a selected symptoms
RETURN

  EXAMPLE

  SEE ALSO


/*****************************************************************************/

void RefreshSelectSymptom ( )
{
VBOEN_OBJECT     *_ptr, _spt1, *_object;
int                 _end, _i, _x, _j, _max, _temp_max;


      /*****************************************************************/
      /* On doit regarder si le symptome courant est un symptome selectione */
      /*                                                               */
      /* Si il n'est pas selectionne :   on attribue un numer o ce symptome      */
      /* Si il est selectionn,       :   on attribue un numeo aux spts. Selectionnes*/
      /*****************************************************************/


      _spt1.Type = OBJECT_VBOEN_SYMPTOM;
      _spt1.Id   = (long) WinViewBoen->CurrentSpt;
      _object    = (VBOEN_OBJECT *) NULL;
      _object    = GetViewBoenObjectInList ( WinViewBoen->Select, &_spt1 );
      if ( !_object || WinViewBoen->Select == NULL) {

            /*****************************************************************/
            /* On attribue un numero au current car il n'est pas dans les selectionnes */
            /*****************************************************************/

            RefreshButton (WinViewBoen->SCNumber,WinViewBoen->CurrentSpt);
            /*****************************************************************/
            /*  REFRESH de la window */
            /*****************************************************************/

         /* FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_DISPLAY );
            DisplayViewBoenWindow ( WinViewBoen->Win, VBOE N_DUMMY_REFRESH );*/

      }
      else {


            /***********************************************/
            /* On compte le nombre de symptomes a attribuer */
            /***********************************************/
            _i   =  1;
            _ptr =  WinViewBoen->Select;
            _max = -1;
            _j   =  0;
            while ( !_j ) {
                        if ( (int)_ptr->Id > _max ) _max = (int) _ptr->Id;
                        if ( _ptr->Next != NULL )   {
                                                      _ptr = _ptr -> Next;
                                                    _i++;
                                                }
                        else _j = 1;
                     }


            _ptr = WinViewBoen->Select;

            _temp_max =  _max;
            _end      =  0;
            while ( _end < _i ) {

      RefreshButton (WinViewBoen->SCNumber,_temp_max);
```

```
        _max = _temp_max;
        _end ++;

        _ptr      =  WinViewBoen->Select;
        _temp_max = -1;

        _x = 0;
        while ( !_x ) {
               if ( ((int)_ptr->Id > _temp_max) && ((int)_ptr->Id < _max) )
                      _temp_max = (int)_ptr->Id;

               if (_ptr->Next != NULL) _ptr =  _ptr->Next;
               else                    _x = 1;
        }
               }
               /*****************************************/
               /* On doit liberer les objects selectionnes */
               /*****************************************/
              /*FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_NO_DISPLAY );
              DisplayViewBoenWindow ( WinViewBoen->Win, VBOEN_GO_TO_FIRST ); */

       }
}
```

A.2.2 LA FONCTION X_VBOEN

(événements associes à la fenêtre vboen)

```
A.3 *
       This file was generated by XVT-Design 3.03, a product of:

              XVT Software Inc.
              4900 Pearl East Circle
              Boulder, CO USA 80301
              303-443-4223, fax 303-443-0969

       Generated on Mon Jun 16 08:37:35 1997
*/

#include "xvt.h"
#include "xvtcm.h"
#include "x_radar.h"

/*
       Information about the window
*/
#define WIN_RES_ID W_VBOEN
#define WIN_FLAGS 0x2L
#define WIN_CLASS ""
#define WIN_BORDER W_DOC
#include "radar.h"

/*
       Handler for window W_VBOEN ("Boenninghausen Symptoms")
*/
long XVT_CALLCONV1
#if XVT_CC_PROTO
W_VBOEN_eh XVT_CALLCONV2 (WINDOW xdWindow, EVENT *xdEvent)
```

```
#else
W_VBOEN_eh XVT_CALLCONV2 (xdWindow, xdEvent)
WINDOW xdWindow;
EVENT *xdEvent;
#endif
{
        short xdControlId = xdEvent->v.ctl.id;

        switch (xdEvent->type) {
        case E_CREATE:
                /*
                        Window has been created; first event sent to newly-created
                        window.
                */
                {
                CurrentWindow = xdWindow;
                }
                break;
        case E_DESTROY:
                /*
                        Window has been closed; last event sent to window.
                */
                xdRemoveHelpAssoc( xdWindow );
                {
                }
                break;
        case E_FOCUS:
                {
                }
                break;
        case E_SIZE:
                /*
                        Size of window has been set or changed; sent when window is
                        created or subsequently resized by user or via xvt_vobj_move.
                */
                {
                }
                break;
        case E_UPDATE:
                /*
                        Window requires updating.
                */
                {
                xvt_dwin_clear(xdWindow, (COLOR)xvt_vobj_get_attr(xdWindow, ATTR_BACK_COLOR));
                DisplayViewBoenWindow ( xdWindow, VBOEN_REFRESH );
                }
                break;
        case E_CLOSE:
                /*
                        Request to close window; user operated "close" menu item on
```

```
                window system menu, or operated "close" control on window
                frame. Not sent if Close on File menu is issued. Window not
                closed unless xvt_vobj_destroy is called.
        */
        {
        TerminateViewBoenWindow ( xdWindow );
        }
        break;
case E_CHAR:
        /*
                Character typed.
        */
        {
        ViewBoenKeyboardHandler ( xdWindow, xdEvent );
        }
        break;
case E_MOUSE_UP:
        /*
                Mouse was released
        */
        {
        ViewBoenMouseHandler ( xdWindow, xdEvent );
        }
        break;
case E_MOUSE_DOWN:
        /*
                Mouse was pressed
        */
        {
        ViewBoenMouseHandler ( xdWindow, xdEvent );
        }
        break;
case E_MOUSE_DBL:
        /*
                Mouse was double clicked
        */
        {
        ViewBoenMouseHandler ( xdWindow, xdEvent );
        }
        break;
case E_MOUSE_MOVE:
        /*
                Mouse was moved
        */
        {
        ViewBoenMouseHandler ( xdWindow, xdEvent );
        }
        break;
case E_HSCROLL:
        {
```

```
        /*
                Horizontal scrollbar on frame was operated
        */
        switch (xdEvent->v.scroll.what) {
        case SC_LINE_UP:
                break;
        case SC_LINE_DOWN:
                break;
        case SC_PAGE_UP:
                break;
        case SC_PAGE_DOWN:
                break;
        case SC_THUMB:
                break;
        case SC_THUMBTRACK:
                break;
        default:
                break;
        }
        }
        break;
case E_VSCROLL:
        {
        /*
                Vertical scrollbar on frame was operated
        */
        switch (xdEvent->v.scroll.what) {
        case SC_LINE_UP:
                break;
        case SC_LINE_DOWN:
                break;
        case SC_PAGE_UP:
                break;
        case SC_PAGE_DOWN:
                break;
        case SC_THUMB:
                break;
        case SC_THUMBTRACK:
                break;
        default:
                break;
        }
        }
        break;
case E_COMMAND:
        /*
                User issued command on window menu bar (menu bar at top of
                screen for Mac/CH).
        */
        {
```

```
            switch ( xdEvent->v.cmd.tag ) {
               /*case M_EDIT_CUT:
                    DeleteSymptoms ();
                    break;
               case M_EDIT_PASTE:
                    BPasteSC11ToCurrentSC ();
                    xvt_dwin_invalidate_rect ( xdWindow, NULL );
                    break;
               case M_EDIT_COPY:
                    BCopySptToSC_11 ();
                    break;*/
               default:
                    do_VBOEN_MENUBAR(xdWindow, xdEvent);
                    break;
            }
            }
            break;
      case E_CONTROL:
            /*
                    User operated control in window.
            */
            {

            switch(xdControlId) {
            case W_VBOEN_SC_LOC: /* "&Loc" */
                    {
                    ChangeViewBoenClipboard ( xdWindow, (int) 0 );
                    }
                    break;
            case W_VBOEN_SC_SENS: /* "&Sens" */
                    {
                    ChangeViewBoenClipboard ( xdWindow, (int)1 );
                    }
                    break;
            case W_VBOEN_SC_MOD: /* "&Mod" */
                    {
                    ChangeViewBoenClipboard ( xdWindow, (int)2 );
                    }
                    break;
            case W_VBOEN_SC_CONC: /* "&Conc" */
                    {
                    ChangeViewBoenClipboard ( xdWindow, (int)3 );
                    }
                    break;
            case W_VBOEN_ANALYS: /* "Anal&ys" */
                    {
                    unsigned char _sentense [MSG_MAX_TEXT_LENGTH +1],
                                  _default [MSG_MAX_TEXT_LENGTH +1],
                                  _label2 [MSG_MAX_TEXT_LENGTH +1];
                    int _i;
```

```
                xvt_win_set_cursor(TASK_WIN, CURSOR_WAIT);
                DbVirtualClose(2);
                if ( GoToAnalyse()){

                  WinViewBoen->StartAnalysis = TRUE;
                  TerminateViewBoenWindow ( xdWindow );
                }
                else{
                        /*************************/
                        /* On demande confirmation */
                        /*************************/

                        GetMessage ( FileMessage, 65, Option.General.Lang, &RecMessage );
                        strcpy ( _default, RecMessage.MsgText );
                        GetMessage ( FileMessage, 66, Option.General.Lang, &RecMessage );
                        strcpy ( _label2, RecMessage.MsgText );


                        GetMessage ( FileMessage, 4037, Option.General.Lang, &RecMessage );
                        sprintf ( _sentense, RecMessage.MsgText);

                        if ( xvt_dm_post_ask( _default, _label2, NULL, _sentense )!= RESP _DEFAULT
) {


                                CheckCompleteSymptom();
                                for (_i=0;_i<9;_i++){
                                        if( (! CC[_i].Complet) && (CC[_i].Present)) {
                                                InitializeViewCheckWindow();
                                                break;
                                        }
                                }

                        }
                        else{

                                WinViewBoen->StartAnalysis = TRUE;
                                TerminateViewBoenWindow ( xdWindow );


                        }

                //xvt_dm_post_note("Nbre de symptoms insuffisant");
                }
                }
                break;
        case W_VBOEN_CHECK: /* "Chec&k" */
                {
                int _i;
                CheckCompleteSymptom();
                for (_i=0;_i<9;_i++)
```

```
        {
         if( (! CC[_i].Complet) && (CC[_i].Present))
           {
            InitializeViewCheckWindow();
            break;
            }
         }
        if (_i==9) xvt_dm_post_note( GetMessage ( FileMessage, 4029,
                                    Option.General.Lang, &RecMessage) );

        }
        break;
case W_VBOEN_OK: /* "C&lose" */
        {
        /*
        ! 7017
        */
        TerminateViewBoenWindow ( xdWindow );
        }
        break;
case W_VBOEN_HELP: /* "&Help" */
        {
        RadarHelp ( xdWindow, (int) 003 );
        }
        break;
case W_VBOEN_VSCROLL:                    {
        /*
                Vertical scrollbar control was operated
        */
        long      _pos;




        switch (xdEvent->v.ctl.ci.v.scroll.what) {
        case SC_LINE_UP:
                        ChangeViewBoenLine ( xdWindow, VBOEN_LINE_UP );
                break;
        case SC_LINE_DOWN:
                        ChangeViewBoenLine ( xdWindow, VBOEN_LINE_DOWN );
                break;
        case SC_PAGE_UP:
                        DisplayViewBoenWindow (xdWindow, VBOEN_PAGE_UP);
                break;
        case SC_PAGE_DOWN:
                        DisplayViewBoenWindow (xdWindow, VBOEN_PAGE_DOWN);
                break;
        case SC_THUMB:
```

```
                _pos = GetWindowScrollBarThumbPos (
                        xvt_win_get_ctl ( xdWindow, W_VBOEN_VSCROLL),
                        HVSCROLL, 1L,
                        SC[WinViewBoen->SCNumber].NbSpt,
                        (int)xdEvent->v.ctl.ci.v.scroll.pos );
                if ( _pos == 1l )    _pos = 0l;
                if ( _pos <   0l )    _pos = 0l;
                        if ( _pos >   (long) SC[WinViewBoen->SCNumber].NbSpt ) {
                                _pos = (long) SC[WinViewBoen->SCNumber].NbSpt;
                        }
                SetWindowScrollBar ( xvt_win_get_ctl ( xdWindow, W_VBOEN_VSCROLL ),
                        HVSCROLL, 1L,
                        SC[WinViewBoen->SCNumber].NbSpt,
                        _pos );

                        WinViewBoen->CurrentSpt = (int) _pos;
                        WinViewBoen->FirstSpt      = (int) _pos;
                        DisplayViewBoenWindow (xdWindow, VBOEN_DUMMY_REFRESH );

                        break;
        case SC_THUMBTRACK:

                _pos = GetWindowScrollBarThumbPos (
                        xvt_win_get_ctl ( xdWindow, W_VBOEN_VSCROLL),
                        HVSCROLL, 1L,
                        SC[WinViewBoen->SCNumber].NbSpt,
                        (int)xdEvent->v.ctl.ci.v.scroll.pos );
                if ( _pos == 1l )    _pos = 0l;
                if ( _pos <   0l )    _pos = 0l;
                        if ( _pos >   (long) SC[WinViewBoen->SCNumber].NbSpt ) {
                                _pos = (long) SC[WinViewBoen->SCNumber].NbSpt;
                        }
                SetWindowScrollBar ( xvt_win_get_ctl ( xdWindow, W_VBOEN_VSCROLL ),
                        HVSCROLL, 1L,
                        SC[WinViewBoen->SCNumber].NbSpt,
                        _pos );

                        WinViewBoen->CurrentSpt = (int) _pos;
                        WinViewBoen->FirstSpt      = (int) _pos;
                        DisplayViewBoenWindow (xdWindow, VBOEN_DUMMY_REFRESH );
                        break;
        }
        }
        break;
case W_VBOEN_CASE: /* "C&ase" */
        {
         CalculateSymptomCase();
         ChangeViewBoenClipboard ( xdWindow, (int)4 );
        }
        break;
```

```
                default:
                        break;
                }
                }
                break;
        case E_FONT:
                /*
                        User issued font command on window menu bar (menu bar at top of
                        screen for Mac/CH).
                */
                {
                }
                break;
        case E_TIMER:
                /*
                        Timer associated with window went off.
                */
                {
                }
                break;
        case E_USER:
                /*
                        Application initiated.
                */
                {
                ManageViewBoenUserEvent ( xdWindow, xdEvent );
                }
                break;
        default:
                break;
        }
        xvt_tx_process_event(xdWindow, xdEvent);
        return 0L;
}
```

A.3.1.1 LA FONCTION X_MVBOEN

Gère les événements du menu associé à la fenêtre Vboen.

```
/*
        This file was generated by XVT-Design 3.03, a product of:

                XVT Software Inc.
                4900 Pearl East Circle
                Boulder, CO USA 80301
                303-443-4223, fax 303-443-0969

        Generated on Mon May 12 23:50:54 1997
*/

#include "xvt.h"
#include "xvtcm.h"
```

```
#include "x_radar.h"
#include "radar.h"

/*
        This function is the menu event handler for menu VBOEN_MENUBAR
*/

void
#if XVT_CC_PROTO
do_VBOEN_MENUBAR (WINDOW xdWindow, EVENT *xdEvent)
#else
do_VBOEN_MENUBAR (xdWindow, xdEvent)
WINDOW xdWindow;
EVENT *xdEvent;
#endif
{
        MENU_TAG tag = xdEvent->v.cmd.tag;

        switch (tag) {
        case _M_VBOEN_FILE_IMPORT: /* Menu "Import" */
                {
                MenuCallImportFromChip ();
                }
                break;
        case _M_VBOEN_FILE_EXPORT: /* Menu "Export" */
                {
                MenuCallExportToChip ();
                }
                break;
        case _M_VBOEN_FILE_SAVE_REPERT: /* Menu "Save to patient file" */
                {
                CallSaveCase(SAVECASE_PATIENT_SAVE);
                }
                break;
        case _M_VBOEN_FILE_RECALL_REPERT: /* Menu "Recall from patient file" */
                {
                CallRecallCase(SAVECASE_PATIENT_RECALL_BOEN);
                }
                break;
        case _M_VBOEN_FILE_DELETE_REPERT: /* Menu "Delete from patient file" */
                {
                CallDeleteCase(SAVECASE_PATIENT_DELETE);
                }
                break;
        case _M_VBOEN_DELETE: /* Menu "Delete symptoms" */
                {
                BDeleteSymptoms();
                }
                break;
        case _M_VBOEN_EDIT_SET_CSN: /* Menu "Give a number" */
```

```
        {

        if (!xvt_win_create_res(W_NSYMP, TASK_WIN, EM_ALL, W_NSYMP_eh, 0L))
                xvt_dm_post_error("Can't open window");
        }
        break;
case _M_VBOEN_EDIT_RESET_ALL_CSN: /* Menu "Delete all "  */
        {
        /*FreeSptInMemory ( -1, 1 );
        WinViewBoen->FirstSpt = 0;
        WinViewBoen->LastSpt = 0;
        DeleteSOSCase ( "SOS" );
        DisplayViewBoenWindow ( xdWindow, VBOEN_REFRESH );*/
        DeleteNumberSymptom();
        }
        break;
case _M_VBOEN_EDIT_RESET_CURR: /* Menu "Delete a number" */
        {


        if (!xvt_win_create_res(W_DNUM, TASK_WIN, EM_ALL, W_DNUM_eh, 0L))
                xvt_dm_post_error("Can't open window");
        }
        break;
case _M_VBOEN_EDIT_SORT: /* Menu "Clinical case" */
        {
        InitializeViewCliniqueWindow();
        /*if (!xvt_win_create_res(WIN_CCASE, TASK_WIN, EM_ALL, WIN_CCASE_eh, 0L))
                xvt_dm_post_error("Can't open window");*/
        }
        break;
case _M_VBOEN_MOVE_TOP: /* Menu "Top" */
        {
        BMoveCurrentSptTop ( WinViewBoen->SCNumber, WinViewBoen->CurrentSpt );
        }
        break;
case _M_VBOEN_MOVE_BOTTOM: /* Menu "Bottom" */
        {
        BMoveCurrentSptBottom ( WinViewBoen->SCNumber, WinViewBoen->CurrentSpt );
        }
        break;
case _M_VBOEN_MOVE_UP: /* Menu "Up" */
        {
        BMoveCurrentSptUp ( WinViewBoen->SCNumber, WinViewBoen->CurrentSpt );
        }
        break;
case _M_VBOEN_MOVE_DOWN: /* Menu "Down" */
        {
        BMoveCurrentSptDown ( WinViewBoen->SCNumber, WinViewBoen->CurrentSpt );
        }
```

```
                break;
        default:
                {
                xvt_win_dispatch_event(TASK_WIN, xdEvent);
                }
                break;
        }
}
```

A.3.2 MODULE CHECK

```
/*****************************************************************************

        SOURCE FILE
                CHECK.C

        MODULE
                View Checken Symptoms window

        AUTHOR
              Diagne Papa Moussa            26/09/96

        DESCRIPTION
                Contain all functions for the management of the
                View Checken Symptoms window


*****************************************************************************/


/*---- INCLUDE SECTION ----------------------------------------------------*/

#include "xvt.h"
#include "x_radar.h"
#include "radar.h"
#include "math.h"
/*---- DEFINE SECTION -----------------------------------------------------*/


/*---- VARIABLE SECTION ---------------------------------------------------*/




/*---- LOW LEVEL FUNCTIONS PROTOTYPES -------------------------------------*/


/*---- HIGH FUNCTION SECTION ----------------------------------------------*/
```

```
/*************************************************************************
Cette fonction la structure donnée associée au check
*************************************************************************/


void InitCheck()
        { int _i;
          for(_i=0; _i<16; _i++)
          {
           CC[_i].Present = 0;
           CC[_i].Complet = 0;
           CC[_i].Check   = 0;
          }
          FreeSptInMemory(4,0);
        }


/*************************************************************************/
Cette fonction analyse le champ Check et affiche les
caractéristiques faisant défaut
/*************************************************************************/
 void CheckInMemory(_SCNumber)
 int _SCNumber;

 {
  if(CC[_SCNumber].Present == 1)
    {
        if( ((CC[_SCNumber].Check) & (LOCALISATION)) == 0)
         {
         xvt_dm_post_note (" Symptom %d Incomplet\n Localisation Inexistante", _SCNumber+1);}
        if ( ((CC[_SCNumber].Check) & (SENSATION)) == 0 )
         {

          xvt_dm_post_note (" Symptom %d Incomplet\n Sensation Inexistante", _SCNumber+1);}
        if ( ((CC[_SCNumber].Check) & (MODALITE)) == 0  )
         {

          xvt_dm_post_note (" Symptom %d Incomplet\n Modalite Inexistante", _SCNumber+1);}
        if (  ((CC[_SCNumber].Check) & (CONCOMITTANT)) == 0 )
         {
          xvt_dm_post_note (" Symptom %d Incomplet\n Concomittant Inexistante", _SCNumber+1);}
    }
  }


/*********************************************************************/
Cette fonction reçoit un numéro de clipboard, et renvoit la nature du clipboard
S'il s'agit d'une localisation, d'une sensation, d'une modalité, d'un concomitant.
/*********************************************************************/
```

```
unsigned char  CheckType(_SCNumber)
int _SCNumber;
{
 unsigned char _type = 0;

 if(_SCNumber == 0)
      _type= LOCALISATION ;
 if(_SCNumber == 1)
      _type= SENSATION;
 if(_SCNumber == 2)
      _type= MODALITE;
 if(_SCNumber == 3)
      _type= CONCOMITTANT;

 return _type;

 }


/*************************************************************************/
Cette fonction parcourt les clipboards, et remplit notre structure
COMPLET_SYMPTOM
/*************************************************************************/

void  CheckCompleteSymptom()

  { int _SCNumber, num;
    SYMPTOM_IN_MEMORY *_Spt;
    unsigned char c1=0;
    _Spt = (SYMPTOM_IN_MEMORY *)NULL;
    InitCheck();
    for ( _SCNumber = 0; _SCNumber < 3; _SCNumber++)
                                                    /* on fait le check
                                                    seulement dans les trois
                                                    premiers (0,1,2) pour verifier
                                                    les symptomes de la maladie*/
       {
         if ( SC[_SCNumber].First != NULL)
            {
              _Spt = SC[_SCNumber].First;
              do
              {
                c1 = _Spt->Qualif;
                for ( num = 1; c1 != 0; c1 >>= 1)
                {
                 if ( c1 & 1)
                  {
                   CC[num-1].Present = 1;
                   CC[num-1].Check  |= CheckType(_SCNumber);
```

```
                    if( ((CC[num-1].Check) & (LOCALISATION)) &&
                        ((CC[num-1].Check) & (SENSATION)) &&
                        ((CC[num-1].Check) & (MODALITE)) /* &&
                        ((CC[num-1].Check) & (CONCOMITTANT))*/ )
                        CC[num-1].Complet = 1;
                     else  CC[num-1].Complet = 0;

                    }
                    num++;
                  }
                /*  c1 = _Spt->Degrees;
                  for ( num = 1; c1 != 0; c1 >>= 1)
                  {
                   if ( c1 & 1)
                    {
                     CC[num+7].Present = 1;
                     CC[num+7].Check  |= CheckType(_SCNumber);
                      if( ((CC[num+7].Check) & (LOCALISATION)) &&
                          ((CC[num+7].Check) & (SENSATION)) &&
                          ((CC[num+7].Check) & (MODALITE))  &&
                          ((CC[num+7].Check) & (CONCOMITTANT)) )
                         CC[num+7].Complet = 1;
                       else  CC[num+7].Complet = 0;

                    }
                   num++;
                  }*/
                  _Spt = _Spt->Next;
                 }
                 while ( _Spt != NULL);
                }
        }
   }


/*********************************************************************/
Cette fonction reçoit un numéro de symptôme complet, copie dans un clipoard tampon.
Les symptômes constituant le cas complet.
/*********************************************************************/
void  DisplayCompleteSymptom(num)
  int num;
    {
     int _SCNumber;
     unsigned char c1;
     SYMPTOM_IN_MEMORY *_Spt;
     FreeSptInMemory(4,0);

     for(_SCNumber=0; _SCNumber<4; _SCNumber++)
     {
      if (SC[_SCNumber].First !=NULL)
```

```
    {
     if (!(_Spt=(SYMPTOM_IN_MEMORY *) Malloc ( sizeof(SYMPTOM_IN_MEMORY))))
     xvt_dm_post_note("Erreur d allocation");
     else
     _Spt->Next = (SYMPTOM_IN_MEMORY *)NULL;
    _Spt = SC[_SCNumber].First;
            do
            {
              if (num >0 && num <9)
                 {
                  c1 = 1;
                  c1 = c1 << (num-1);
                  if( _Spt->Qualif & c1)
                    CopySptToSC(4,_SCNumber,_Spt);

                    /*******************************
                    modifier aprŠs RV avec coquillard
                    on affiche aussi les concommitants
                    associ,s au numero de symptome
                    complet
                    ********************************/

                  if( _Spt->Degrees & c1)
                     CopySptToSC(4,_SCNumber,_Spt);

                  }
              else
               if ( num >8 && num <17)
                 {
                  c1 = 1;
                  c1 = c1 << (num-9);
                  if( _Spt->Degrees & c1)
                     CopySptToSC(4,_SCNumber,_Spt);

                  }
              _Spt = _Spt->Next;
             }
             while ( _Spt != NULL);

    }
   }
  }
/*****************************************************************/
Cette fonction prépare l'analyse, regroupe les symptômes selon
les caractéristiques connues, et enregistre le tout dans un clipboard tampon.
/*****************************************************************/
void VboenAnalyse()
   {
    int _SCNumber, num;
    unsigned char c1,c2;
```

```
SYMPTOM_IN_MEMORY *_Spt;
FreeSptInMemory(4,0);

for(_SCNumber=0; _SCNumber<4; _SCNumber++)
{
 if (SC[_SCNumber].First !=NULL)
 {
  if (!(_Spt=(SYMPTOM_IN_MEMORY *) Malloc ( sizeof(SYMPTOM_IN_MEMORY))))
  xvt_dm_post_note("Erreur d allocation");
  else
  _Spt->Next = (SYMPTOM_IN_MEMORY *)NULL;
 _Spt = SC[_SCNumber].First;
         do
         {
           c1 = _Spt->Qualif;

           for ( num = 1; c1 != 0; c1 >>= 1)
           {
            if ( c1 & 1)
             {
              _Spt->Group = 'A'+(num-1);
              CopySptToSC_5(_Spt);
              }
              num++;
            }

           c2 = _Spt->Degrees;

           for ( num = 1; c2 != 0; c2 >>= 1)
           {
            if ( c2 & 1)
             {
              /*******************************
                aprŠs RV avec Coqu
              on prend en compte dans le m^me groupe les
              concommitants
              ****************************************/
              /*_Spt->Group = 'A'+(num+7); */
              _Spt->Group = 'A'+(num-1);
              CopySptToSC_5(_Spt);
              }
             num++;
            }
            /*if((c1 == 0) && (c2==0))
                  break;*/
           _Spt = _Spt->Next;
         }
         while ( _Spt != NULL);

}
```

```
    }
   }

/*****************************************************************/
Cette fonction prépare l'analyse, regroupe les symptômes selon
les caractéristiques connues, et enregistre le tout dans un clipboard tampon.

/*****************************************************************/
void NewVboenAnalyse()
  {
    int _SCNumber;
    SYMPTOM_IN_MEMORY *_Spt;

    FreeSptInMemory(4,0);

    for(_SCNumber=0; _SCNumber<4; _SCNumber++)
    {
      if ( SC[_SCNumber].First !=NULL)
       {
         if ( !(_Spt =
              (SYMPTOM_IN_MEMORY *) Malloc (sizeof (SYMPTOM_IN_MEMORY)) ) ) {
              xvt_dm_post_note( GetMessage ( FileMessage, 84, Option.General.Lang, &RecMessage) );
              return ( RADAR_NOT_OK );
           }

         _Spt->Next = (SYMPTOM_IN_MEMORY *)NULL;
         _Spt = SC[_SCNumber].First;
               do
               {
                  if ( (_Spt->Qualif !=0)
                       || (_Spt->Degrees !=0)){
                    _Spt->Group = 'A'+(_SCNumber);
                    CopySptToSC_5(_Spt);
                   }
                  _Spt = _Spt->Next;
                }
                while ( _Spt != NULL);

       }
     }
  }

/*****************************************************************/
```

```
Copie un symptôme dans un clipboard tampon.
/*********************************************************/
int CopySptToSC_5 ( _spt )

 SYMPTOM_IN_MEMORY *_spt;


{

int                  _i,
                     _j;


            /*********************************************************/
            /* Allocating a structure to take this symptom in memory */
            /*********************************************************/
            if ( !(SptInMemory =
               (SYMPTOM_IN_MEMORY *) Malloc (sizeof (SYMPTOM_IN_MEMORY)) ) ) {
               xvt_dm_post_note( GetMessage ( FileMessage, 84, Option.General.Lang, &RecMessage) );
               return ( RADAR_NOT_OK );
            }
            SptInMemory->Next = (SYMPTOM_IN_MEMORY *)NULL;
#if DEBUG
            SptInMemory->Check = DB_CHECK;
#endif
            /*********************************************/
            /* Allocation of memory for the symptom texts */
                              /*********************************************/
            if ( !(SptInMemory->SptTextLg1 =
               (unsigned char *) Malloc ( strlen (_spt->SptTextLg1) + 2 )) )    {
               xvt_dm_post_note( GetMessage ( FileMessage, 88, Option.General.Lang, &RecMessage) );
               Free ( (unsigned char *) SptInMemory);
               return (RADAR_NOT_OK);
            }

            if ( !(SptInMemory->SptTextLg2 =
               (unsigned char *) Malloc ( strlen (_spt->SptTextLg2) + 2)) )    {
               xvt_dm_post_note( GetMessage ( FileMessage, 89, Option.General.Lang, &RecMessage) );
               Free ( (unsigned char *) SptInMemory->SptTextLg1);
               Free ( (unsigned char *) SptInMemory);
               return (RADAR_NOT_OK);
            }

            memset (SptInMemory->SptTextLg1, '\0', strlen (_spt->SptTextLg1));
            memset (SptInMemory->SptTextLg2, '\0', strlen (_spt->SptTextLg2));

            /*********************************************/
            /* On alloue de la memoire pour les remedes   */
            /*********************************************/
```

```
        _i =  (int) _spt->NbRemedies;
        _i += 2;

        if ( !(SptInMemory->RmdList =
             (REMED_LIST *) Malloc ( (sizeof (REMED_LIST) * (_i)) ))  )   {
           xvt_dm_post_note( GetMessage ( FileMessage, 90, Option.General.Lang, &RecMessage) );
           Free ( (unsigned char *) SptInMemory->SptTextLg1);
           Free ( (unsigned char *) SptInMemory->SptTextLg2);
           Free ( (unsigned char *) SptInMemory);
           return (RADAR_NOT_OK);
        }

        /*******************************************/
        /* On recopie le symptome membres … membres */
        /*******************************************/
        SptInMemory->Intensity =   (char) 1;
        /*SptInMemory->Intensity =_spt->Intensity ;*/
        SptInMemory->Group     =   _spt->Group     ;
        SptInMemory->Degrees   =  (unsigned char)  0;
        SptInMemory->Degrees  |= ( unsigned char) 1;
        SptInMemory->Degrees  |= ( unsigned char) 2;
        SptInMemory->Degrees  |= ( unsigned char) 4;
        SptInMemory->Degrees  |= ( unsigned char) 8;


        SptInMemory->Qualif =   ( unsigned char) 0;
        SptInMemory->View =        _spt->View      ;
        SptInMemory->NbRemedies =  _spt->NbRemedies;
        SptInMemory->Origin =      _spt->Origin    ;
        SptInMemory->Langue[0] =   _spt->Langue[0] ;
        SptInMemory->Langue[1] =   _spt->Langue[1] ;

        strcpy( SptInMemory->SptTextLg1, _spt->SptTextLg1 );
        strcpy( SptInMemory->SptTextLg2, _spt->SptTextLg2 );

        for (_j=0; _j<(int)_spt->NbRemedies; _j++) {
           SptInMemory->RmdList[_j].Check    = DB_CHECK;
           SptInMemory->RmdList[_j].RemedId  = (unsigned short) _spt- >RmdList [_j].RemedId;
           SptInMemory->RmdList[_j].Degree   = (unsigned char)  _spt->RmdList [_j].Degree;
           SptInMemory->RmdList[_j].AuthorId = (unsigned short) _spt->RmdList [_j].AuthorId;
        }
        SptInMemory->Next = (SYMPTOM_IN_MEMORY *)NULL;

        UpdateSymptomClipboard ( 5, SptInMemory->NbRemedies );

         return ( RADAR_OK );
}


/*********************************************************/
```

```
Cete fonction sert à lancer l'analyse. Vérifie toutefois
Si les numéros de symptômes complets présents sont complets.
Compte le numéro de symptômes complets. En effet on a fixé un
Minimum de 3 symptômes complets pour lancer l'analyse
/*********************************************************/
int GoToAnalyse()
   {
     int _SCNumber;
     int cpte = 0;
     CheckCompleteSymptom();
     for (_SCNumber = 0; _SCNumber < MAX_NB_CC; _SCNumber++) {
          if(CC[_SCNumber].Present == 1) {

               if( ((CC[_SCNumber].Check) & (LOCALISATION)) &&
                   ((CC[_SCNumber].Check) & (SENSATION)) &&
                   ((CC[_SCNumber].Check) & (MODALITE)) /*&&
                   ((CC[_SCNumber].Check) & (CONCOMITTANT))*/ )
                    cpte++;
               }
          }

          if (cpte >= 1) {

               /*********************************
               on fait l'analyse avec la nouvelle fonction NewVboenAnalyse
               aprŠs Rv avec Coquillard
               *********************************/
                /*VboenAnalyse();*/
                NewVboenAnalyse();
                return cpte ;
                }
       else {

               /*********************************
               on fait l'analyse avec la nouvelle fonction NewVboenAnalyse
               aprŠs Rv avec Coquillard, le syst,me ne doit plus etre fige
               poser la question si on veut ou pas continuer l'analyse
               meme si le cas n'est pas complet, au medecin de decider
               *********************************/
                /*VboenAnalyse();*/
                NewVboenAnalyse();
                return 0;
       }
   }
```

```
/*****************************************************************************
```

```
Cette fonction crée et initialise la fenêtre check symptom
*****************************************************************************/
void
InitializeViewCheckWindow ( )
{
WINDOW  _check;
WIN_DEF *_windef;
RCT *_prct;

        xvt_win_set_cursor ( TASK_WIN, CURSOR_WAIT );
        DbVirtualClose ( 2 );

        if ( ! ( WinViewCheck = (CHECK_WINDOW *)Malloc ( sizeof ( CHECK_WINDOW ) + 1 ) ) ) {
                xvt_dm_post_note( GetMessage ( FileMessage, 104, Option.General.Lang, &RecMessage) );
                xvt_win_set_cursor ( TASK_WIN, CURSOR_ARROW );
                return;
        }

        WinViewCheck->Type              = WINDOW_TYPE_CHECK;

        DbVirtualClose ( 2 );

        /* Create and calculate window */

         if ( ! ( _windef = xvt_res_get_win_def ( W_COMPSYMP ) ) ) {
                xvt_dm_post_note( GetMessage ( FileMessage, 31, Option.General.Lang, &RecMessage ) );
                Free ( (char *)WinViewCheck );
                xvt_win_set_cursor ( TASK_WIN, CURSOR_ARROW );
                return;
         }


         if ( !( _check = xvt_win_create_def ( _windef, TASK_WIN, EM_ALL,

W_COMPSYMP_eh, (long)WinViewCheck ) ) ) {
                xvt_dm_post_note( GetMessage ( FileMessage, 31, Option.General.Lang, &RecMessage ) );
                Free ( (char *)WinViewCheck );
                xvt_win_set_cursor ( TASK_WIN, CURSOR_ARROW );
                return;
         }

         xvt_res_free_win_def ( _windef );

        WinViewCheck->Win               = _check;
        /* Initialize Focus */
        WinViewCheck->Focus             = 0;
        WinViewCheck->SCNumber          = 0;
        WinViewCheck->DisplayNumber     = 0;

                while (WinViewCheck->SCNumber<9)
                 {
```

```
                if(DisplayCheckWindow(WinViewCheck->SCNumber))
                  {
                   WinViewCheck->DisplayNumber = WinViewCheck->SCNumber;
                   break;
                  }
                WinViewCheck->SCNumber++;
                }



}
/*********************************************************/
Cette fonction garnit la fenêtre check symptom.
/*********************************************************/
void   UpdateCheckWindow( win)
WINDOW win;
{
              short      _h,
                         _v;
               RCT rct1;
               CBRUSH brush;
               CPEN   pen;

               xvt_dwin_clear ( win, Option.Patient.ColorBg ) ;
               xvt_dwin_set_fore_color ( win, COLOR_BLACK ) ;


  pen.color=Option.Patient.ColorBg;
  pen.pat=PAT_SOLID;
  pen.style=P_SOLID;
  pen.width=1;
  xvt_dwin_set_cpen ( win, &pen ) ;
  brush.pat=PAT_SOLID;
  brush.color=Option.Patient.ColorBg;
  xvt_dwin_set_cbrush ( win, &brush ) ;
  xvt_rect_set(&rct1,11,238,284,287);
  xvt_dwin_draw_rect ( win, &rct1 ) ;
  DrawSunkenBox ( win, &rct1 ) ;
  xvt_rect_set(&rct1,285,36,398,236);
  xvt_dwin_draw_rect ( win, &rct1 ) ;
  DrawSunkenBox ( win, &rct1 ) ;

  /*_v = 70;
  WindowChangeResolution ( &_v, WINDOW_VERT, RESOLUTION_UP );

  _h = 15;
  WindowChangeResolution ( &_h, WINDOW_HOR, RESOLUTION_UP );
  xvt_dwin_draw_text ( win, _h, _v, GetMessage ( FileMessage, 4024, Option.General.Lang, &RecMessage ),
-1 );*/
```

```
  }


/*********************************************************************/
Cette fonction reçoit un numéro de symptôme complet, vérifie s'il est incomplet,
Envoie 1 s'il est incomplet, et 0 sinon
Et affiche les caractéristiques manquantes.
/*********************************************************************/

  int DisplayCheckWindow(_SCNumber)
  int _SCNumber;
{
 char _buf[100],_buf1[25];


   CheckCompleteSymptom();

  if((CC[_SCNumber].Present == 1) && (CC[_SCNumber].Complet == 0))
    {
        sprintf ( _buf, GetMessage ( FileMessage, 4017, Option.General.Lang, &RecMessage ),
_SCNumber+1);
       DisplayTextEdit(xvt_win_get_tx(WinViewCheck->Win,W_COMPSYMP_TEDIT),_buf);
       sprintf ( _buf1,"\n\n%s", GetMessage ( FileMessage, 4018, Option.General.Lang, &RecMessage ));
       strcat(_buf,_buf1);
       DisplayTextEdit(xvt_win_get_tx(WinViewCheck->Win,W_COMPSYMP_TEDIT),_buf);

       if( ((CC[_SCNumber].Check) & (LOCALISATION)) == 0)
        {
         sprintf ( _buf1,"\n%s", GetMessage ( FileMessage, 4019, Option.General.Lang, &RecMessage ));
         strcat(_buf,_buf1);
         DisplayTextEdit(xvt_win_get_tx(WinViewCheck->Win,W_COMPSYMP_TEDIT),_buf);
         }

       if ( ((CC[_SCNumber].Check) & (SENSATION)) == 0 )
        {
         sprintf ( _buf1,"\n%s", GetMessage ( FileMessage, 4021, Option.General.Lang, &RecMessage ));
         strcat(_buf,_buf1);
         DisplayTextEdit(xvt_win_get_tx(WinViewCheck->Win,W_COMPSYMP_TEDIT),_buf);
         }

       if ( ((CC[_SCNumber].Check) & (MODALITE)) == 0  )
        {
         sprintf ( _buf1,"\n%s", GetMessage ( FileMessage, 4022, Option.General.Lang, &RecMessage ));
         strcat(_buf,_buf1);
         DisplayTextEdit(xvt_win_get_tx(WinViewCheck->Win,W_COMPSYMP_TEDIT),_buf);
         }

      /* if (  ((CC[_SCNumber].Check) & (CONCOMITTANT)) == 0 )
        {
         sprintf ( _buf1,"\n%s", GetMessage ( FileMessage, 4023, Option.General.Lang, &RecMessage ));
         DisplayTextEdit(xvt_win_get_tx(WinViewCheck->Win,W_COMPSYMP_TEDIT),_buf);
```

```
        }*/

        return 1;
      }

    else return 0;
   /* xvt_win_set_cursor ( TASK_WIN, CURSOR_ARROW );*/

  }
```

A.3.2.1 FONCTION X_CPSYMP

Gère les évemenents associés à la fenêtre check complete symptom

```
/*
       This file was generated by XVT-Design 3.03, a product of:

              XVT Software Inc.
              4900 Pearl East Circle
              Boulder, CO USA 80301
              303-443-4223, fax 303-443-0969

       Generated on Mon Mar 24 15:50:45 1997
*/

#include "xvt.h"
#include "xvtcm.h"
#include "x_radar.h"
#include "radar.h"

/*
       Information about the window
*/
#define WIN_RES_ID W_COMPSYMP
#define WIN_FLAGS 0x802L
#define WIN_CLASS ""
#define WIN_BORDER W_DOC

/*
       Handler for window W_COMPSYMP ("Complete Symptom")
*/

long XVT_CALLCONV1
#if XVT_CC_PROTO
W_COMPSYMP_eh XVT_CALLCONV2 (WINDOW xdWindow, EVENT *xdEvent)
#else
W_COMPSYMP_eh XVT_CALLCONV2 (xdWindow, xdEvent)
WINDOW xdWindow;
EVENT *xdEvent;
#endif
```

```
{
        short xdControlId = xdEvent->v.ctl.id;
        static WIN_MSG WinMsgCheck [] = {
                        W_COMPSYMP,                     -1,     '\0',
                        W_COMPSYMP_ANALYS,              623,    '\0',
                        W_COMPSYMP_CANCEL,              856,    '\0',
                        0,                               0,       '\0' };

        switch (xdEvent->type) {
        case E_CREATE:
                /*
                        Window has been created; first event sent to newly-created
                        window.
                */
                {
                 LoadMessagesForWindow(xdWindow,(WIN_MSG*) WinMsgCheck);
                }
                break;
        case E_DESTROY:
                /*
                        Window has been closed; last event sent to window.
                */
                xdRemoveHelpAssoc( xdWindow );
                {
                }
                break;
        case E_FOCUS:
                {
                /*
                        Window has lost or gained focus .
                */
                if (xdEvent->v.active)  {
                        /*
                                Window has gained focus
                        */

                } else {
                        /*
                                Window has lost focus
                        */
                }
                }
                break;
        case E_SIZE:
                /*
                        Size of window has been set or changed; sent when window is
                        created or subsequently resized by user or via xvt_vobj_move.
                */
                {
                }
```

```
        break;
case E_UPDATE:
        /*
                Window requires updating.
        */
        {
        UpdateCheckWindow(xdWindow);

        }
        break;
case E_CLOSE:
        /*
                Request to close window; user operated "close" menu item on
                window system menu, or operated "close" control on window
                frame. Not sent if Close on File menu is issued. Window not
                closed unless xvt_vobj_destroy is called.
        */
        {
        xvt_vobj_destroy(xdWindow);
        }
        break;
case E_CHAR:
        /*
                Character typed.
        */
        {
        }
        break;
case E_MOUSE_UP:
        /*
                Mouse was released
        */
        {
        }
        break;
case E_MOUSE_DOWN:
        /*
                Mouse was pressed
        */
        {
        }
        break;
case E_MOUSE_DBL:
        /*
                Mouse was double clicked
        */
        {
        }
        break;
case E_MOUSE_MOVE:
```

```
        /*
                Mouse was moved
        */
        {
        }
        break;
case E_HSCROLL:
        {
        /*
                Horizontal scrollbar on frame was operated
        */
        switch (xdEvent->v.scroll.what) {
        case SC_LINE_UP:
                break;
        case SC_LINE_DOWN:
                break;
        case SC_PAGE_UP:
                break;
        case SC_PAGE_DOWN:
                break;
        case SC_THUMB:
                break;
        case SC_THUMBTRACK:
                break;
        default:
                break;
        }
        }
        break;
case E_VSCROLL:
        {
        /*
                Vertical scrollbar on frame was operated
        */
        switch (xdEvent->v.scroll.what) {
        case SC_LINE_UP:
                break;
        case SC_LINE_DOWN:
                break;
        case SC_PAGE_UP:
                break;
        case SC_PAGE_DOWN:
                break;
        case SC_THUMB:
                break;
        case SC_THUMBTRACK:
                break;
        default:
                break;
        }
```

```
            }
            break;
        case E_COMMAND:
            /*
                User issued command on window menu bar (menu bar at top of
                screen for Mac/CH).
            */
            {
            /*
                No menubar was associated with this window
            */
            }
            break;
        case E_CONTROL:
            /*
                User operated control in window.
            */
            {

            switch(xdControlId) {
            case W_COMPSYMP_PREV: /* "<<" */
                {
                WinViewCheck->SCNumber = WinViewCheck->DisplayNumber-1;
                while ( WinViewCheck->SCNumber>=0)
                {

                   if(DisplayCheckWindow(WinViewCheck->SCNumber))
                     {
                      WinViewCheck->DisplayNumber = WinViewCheck->SCNumber;
                      break;
                     }
                   WinViewCheck->SCNumber= WinViewCheck->SCNumber-1;
                 }
                 if(WinViewCheck->SCNumber < 0) DisplayCheckWindow(WinViewCheck->DisplayNumber);

                }
                break;
            case W_COMPSYMP_NEXT: /* ">>" */
                {
                WinViewCheck->SCNumber = WinViewCheck->DisplayNumber+1 ;
                while ( WinViewCheck->SCNumber<9)
                {
                   if(DisplayCheckWindow(WinViewCheck->SCNumber))
                     {
                      WinViewCheck->DisplayNumber= WinViewCheck->SCNumber;
                      break;
                     }
                   WinViewCheck->SCNumber= WinViewCheck->SCNumber+1;
                 }
                 if(WinViewCheck->SCNumber >= 9) DisplayCheckWindow(WinViewCheck-
>DisplayNumber);
```

```
                }
                break;
        case W_COMPSYMP_CANCEL: /* "Cancel" */
                {
                xvt_vobj_destroy(xdWindow);
                ChangeViewBoenClipboard(WinViewBoen->Win, WinViewBoen->SCNumber );
                }
                break;
        case W_COMPSYMP_ANALYS: /* "Analys" */
                {
                xvt_vobj_destroy(xdWindow);
                xvt_win_set_cursor(TASK_WIN, CURSOR_WAIT);
                DbVirtualClose(2);

                /*****************************
                 le syst,me n'est plus fig,
                 on devrait pouvoir lancer
                 l'analyse apr,s RV apr,s
                 Coquillard
                *****************************
                if ( GoToAnalyse()){
                  WinViewBoen->StartAnalysis = TRUE;
                  TerminateViewBoenWindow ( WinViewBoen->Win );
                }
                else xvt_dm_post_note("Nbre de symptoms insuffisa nt");
                *****************************/

                 GoToAnalyse();
                 WinViewBoen->StartAnalysis = TRUE;
                 TerminateViewBoenWindow ( WinViewBoen->Win );

                }
                break;
        default:
                break;
        }
        }
        break;
case E_FONT:
        /*
                User issued font command on window menu bar (menu bar at top of
                screen for Mac/CH).
        */
        {
        }
        break;
case E_TIMER:
        /*
```

```
                        Timer associated with window went off.
                */
                {
                }
                break;
        case E_USER:
                /*
                        Application initiated.
                */
                {
                switch (xdEvent->v.user.id) {
                case -1:
                default:
                        break;
                }
                }
                break;
        default:
                break;
        }
        xvt_tx_process_event(xdWindow, xdEvent);
        return 0L;
}
```

A.3.3 MODULE VALORISATION

```
/*****************************************************************************/

    SOURCE FILE
        TAKE.C

    MODULE
        Take number symptoms

    AUTHOR
        Diagne Papa Moussa              04/08/96

    DESCRIPTION
        Contain all functions for the management of the
        take number symptoms function

*****************************************************************************/


/*---- INCLUDE SECTION ----------------------------------------------------*/

#include "xvt.h"
#include "x_radar.h"
#include "radar.h"
```

```
/*---- DEFINE SECTION ----------------------------------------------------*/

/*---- VARIABLE SECTION --------------------------------------------------*/

/*****************************************************************************

  NAME
       BDeleteSptInMemory : Deletes a symptom in memory.

  SYNTAX
       void
       BDeleteSptInMemory ( SCNumber, SptNumber );

  PARAMETERS
       int SCNumber                I Clipboard number where to delete spt
       int SptNumber               I Number of the symptom to delete.

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
       Deletes ONE symptom in memory and update the clipboard.

  RETURN


  EXAMPLE

  SEE ALSO

*****************************************************************************/

void
BDeleteSptInMemory ( _SCNumber, _SptNumber)
   int _SCNumber;
   int _SptNumber;

{
SYMPTOM_IN_MEMORY  *_spt,
                   *_old;

int                _i, num;
unsigned char      c1;

      if ((_SCNumber==4)|| (SC[_SCNumber].First == NULL) ) {
      /*xvt_dm_post_note("%d",_SCNumber);
      xvt_dm_post_note( GetMessage ( FileMessage, 109, Option.General.Lang, &RecMessage), _SptNumber
);*/
                return;
```

```
    }


_i   = 0;
_spt = SC[_SCNumber].First;
while ( (_i < _SptNumber) &&
        (_spt->Next != (SYMPTOM_IN_MEMORY *)NULL) ) {
   _old = _spt;
   _spt = _spt->Next;
   _i++;
}
     /************************************************/

if (_i==_SptNumber)
{
     c1 = _spt->Qualif;
     for (num =1; c1 !=0; c1 >>=1)
     {
      if (c1 & 1)
        RectButton[num-1].Button = ON;
        DisplayVBoenButton ( num-1);
        num++;
      }

     c1 = _spt->Degrees;
     for (num =1; c1 !=0; c1>>=1)
      {
       if (c1 & 1)
        RectButton[num+7].Button = ON;
        DisplayVBoenButton ( num+7);
        num++;
       }

   }
/*****************************************************/
/* _old contient le pr,c,dent !!! */
                    /*****************************************************/
                    /* Cas ou on delete le premier symptome en memoire */
                    /*****************************************************/
if ( _i == 0 ) {

   if (_spt->Next == (SYMPTOM_IN_MEMORY *)NULL) {
                           /*****************************************/
                           /* Il n'y a pas d'autres Spt en memoire !! */
                           /*****************************************/
      SC[_SCNumber].First  = (SYMPTOM_IN_MEMORY *)NULL;
      SC[_SCNumber].Last   = (SYMPTOM_IN_MEMORY *)NULL;
      SC[_SCNumber].NbSpt  = 0;
   }
   else {
```

```
                                    /************************************/
                                    /* il y a d'autres symptome apres !! */
                                    /************************************/
        SC[_SCNumber].First  = _spt->Next;
        SC[_SCNumber].NbSpt --;
      }
    }
    else {
                                              /*****************************/
                                              /* Ce n'est pas le premier */
                                              /*****************************/
      if (_spt->Next == (SYMPTOM_IN_MEMORY *)NULL) {
                                         /*******************************/
                                         /* C'est le dernier de la liste */
                                         /*******************************/
        _old->Next = (SYMPTOM_IN_MEMORY *)NULL;
        SC[_SCNumber].Last   = _old;
      }
      else {
                                           /*****************************/
                                           /* Il y en a d'autre apres !! */
                                           /*****************************/
        _old->Next = _spt->Next;
      }
      SC[_SCNumber].NbSpt --;
    }

    /*******************************/
    /* On libere les zones m,moire */
    /*******************************/
    Free ( (unsigned char *) _spt->SptTextLg1);
    Free ( (unsigned char *) _spt->SptTextLg2);
    Free ( (unsigned char *) _spt->RmdList);
    Free ( (unsigned char *) _spt);
    Spt_Not_Saved = 1;

    PictureBarDisplayClip ();

}


/*****************************************************************************

  NAME
       BDeleteSymptoms : Deletes the current or the selected symptoms

  SYNTAX
       void
       DeleteSymptoms ( );
```

```
  PARAMETERS

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
       Makes the dispatching to delete the current symptom (if there are no
          selected symptoms) or to delete the selected symptoms

  RETURN


  EXAMPLE

  SEE ALSO

*****************************************************************************/

void BDeleteSymptoms ( )
{
VBOEN_OBJECT     *_ptr, _spt, *_object;
int                _end, _i, _x, _j, _max, _temp_max;
unsigned char      _sentense [ MSG_MAX_TEXT_LENGTH + 1 ],
                   _default  [ MSG_MAX_TEXT_LENGTH + 1 ],
                   _label2   [ MSG_MAX_TEXT_LENGTH + 1 ],
                   *_ptr3;


   /*************************************************************************/
   /* On doit regarder si le symptome courant est un symptome selectione */
   /*                                                                       */
   /* Si il n'est pas selectionne :      on delete ce symptome          */
   /* Si il est selectionn,       :      on delete les spts. Selectionnes*/
   /*************************************************************************/


   _spt.Type = OBJECT_VBOEN_SYMPTOM;
   _spt.Id   = (long) WinViewBoen->CurrentSpt;
   _object   = (VBOEN_OBJECT *) NULL;
   _object   = GetViewBoenObjectInList ( WinViewBoen->Select, &_spt );

   if ( !_object || WinViewBoen->Select == NULL) {
      /*********************************************************************/
      /* On doit deleter le current car il n'est pas dans les selectionnes */
      /*********************************************************************/

      if (GetConfirmationWindow ( 65, 66, 0, 126 ) != RESP_DEFAULT) {
         return;
      }
```

```
    BDeleteSptInMemory ( WinViewBoen->SCNumber, WinViewBoen->CurrentSpt);

    /*************************************************************************/
    /* Mise a jour de l'etalonnage de la scroll bar et REFRESH de la window */
    /*************************************************************************/

    /*FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_DISPLAY );*/

    if (SC[WinViewBoen->SCNumber].NbSpt > 0) {
       SetWindowScrollBar ( xvt_win_get_ctl ( WinViewBoen->Wi n, W_VBOEN_VSCROLL ),
                            HVSCROLL, 1L, (long) SC[WinViewBoen->SCNumber].NbSpt,
                            (long) WinViewBoen->CurrentSpt );
    }
    /* on delete le dernier de la liste */
    if ( WinViewBoen->CurrentSpt >= (int)SC[WinViewBoen->SCNumber].NbSpt &&
         (int)SC [WinViewBoen->SCNumber].NbSpt > 0) {
       DisplayViewBoenWindow ( WinViewBoen->Win, VBOEN_GO_TO_LAST );
    }
    /* on delete un symptome dans la liste ==> REFRESH */
    else {
       DisplayViewBoenWindow ( WinViewBoen->Win, VBOEN _REFRESH );
    }
}
else {

    /*************************************************************************/
    /* On copie les Spt s,lectionn,s dans le clipboard 11 avant le delete */
    /*************************************************************************/
   /* CopySptToSC_11 ( );*/

    /***********************************************/
    /* On compte le nombre de symptomes a supprimer */
    /***********************************************/
    _i   =  1;
    _ptr =  WinViewBoen->Select;
    _max = -1;
    _j   =  0;
    while ( !_j ) {
       if ( (int)_ptr->Id > _max ) _max = (int) _ptr->Id;

       if ( _ptr->Next != NULL )   {
          _ptr = _ptr -> Next;
          _i++;
       }
       else _j = 1;
    }

    /*************************/
```

```
/* On demande confirmation */
/***************************/
GetMessage ( FileMessage, 65, Option.General.Lang, &RecMessage );
strcpy ( _default, RecMessage.MsgText );
GetMessage ( FileMessage, 66, Option.General.Lang, &RecMessage );
strcpy ( _label2, RecMessage.MsgText );

_ptr3 = NULL;

GetMessage ( FileMessage, 127, Option.General.Lang, &RecMessage );
sprintf ( _sentense, RecMessage.MsgText, _i );

if ( xvt_dm_post_ask( _default, _label2, _ptr3, _sentense )!= RESP_DEFAULT ) {
   return;
}

/*************************************************************************/
/* CAUTION ; You must always delete the greatest Id becaus you           */
/*   don't knw in which order they have been selected, you must          */
/*   always delete the greatest one, like this when you will delete      */
/*   the next one it's Id is still correct.                              */
/*   if you don't delete in decreasing order following the ID's you      */
/*   won't delete the correct one.                                       */
/*************************************************************************/

_ptr = WinViewBoen->Select;

_temp_max =  _max;
_end      =  0;
while ( _end < _i ) {
 BDeleteSptInMemory ( WinViewBoen->SCNumber, _temp_max );

   _max = _temp_max;
   _end ++;

   _ptr      =  WinViewBoen->Select;
   _temp_max = -1;

   _x = 0;
   while ( !_x ) {
      if ( ((int)_ptr->Id > _temp_max) && ((int)_ptr->Id < _max) )
         _temp_max = (int)_ptr->Id;

      if (_ptr->Next != NULL) _ptr =  _ptr->Next;
      else                    _x = 1;
   }
}
/*****************************************/
/* On doit liberer les objects selectionnes */
/*****************************************/
```

```
      /*FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_NO_DISPLAY );*/

      DisplayViewBoenWindow ( WinViewBoen->Win, VBOEN_GO_TO_FIRST );
   }
/*
! 14007
*/
   if ( StillSptInMemory ( ) ) {
      SaveSOSCase ( "SOS" );
   }
   else {
      DeleteSOSCase ( "SOS" );
   }
}
```

```
/*****************************************************************************/

******************************************************************************

  NAME
       GetNumberInMemory : give a number to a symptom in memory.

  SYNTAX
void
GetNumberInMemory ( _SCNumber, _SptNumber, num)

  PARAMETERS
   int num;
   int _SCNumber;
   int _SptNumber;


  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
                GIVE a number to ONE symptom in memory.

  RETURN


  EXAMPLE

  SEE ALSO
```

```
*****************************************************************************/

void
GetNumberInMemory ( _SCNumber, _SptNumber, num)
   int num;
   int _SCNumber;
   int _SptNumber;

{
unsigned char c2;
SYMPTOM_IN_MEMORY  *_spt;


int                _i;

       if ((_SCNumber==4)|| (SC[_SCNumber].First == NULL) ) {
      /*xvt_dm_post_note("%d",_SCNumber);
      xvt_dm_post_note( GetMessage ( FileMessage, 109, Option.General.Lang, &RecMessage), _SptNumber
);*/
               return;
   }

   _i   = 0;
   /*RectButton[num-1].Button = OFF;
   DisplayVBoenButton(num-1);*/
   _spt = SC[_SCNumber].First;
   while ( (_i < _SptNumber) &&
           (_spt != (SYMPTOM_IN_MEMORY *)NULL) ) {
            _spt = _spt->Next;
            _i++;
   }
   if (_i==_SptNumber)
   {
        if ( (num > 0) && (num < 9) && (_SCNumber != 3) )
        {
         c2=1;
         c2 = c2 << (num-1);
        _spt->Qualif |= c2;
        RectButton[num-1].Button = OFF;
        DisplayVBoenButton(num-1);

        }

        else
        if( (num > 8) && (num < 17) &&(_SCNumber == 3) )
        {
          c2=1;
          c2 = c2 << (num-9);
         _spt->Degrees |= c2;
          RectButton[num-1].Button = OFF;
          DisplayVBoenButton(num-1);
```

```
        }

    }

}

/*****************************************************************************

  NAME
                GetNumberSymptom : Give a number to the current or the selected symptoms

  SYNTAX
                void
                GetNumberSymptom();

  PARAMETERS

  INCLUDE
                xvt.h
                radar.h

  DESCRIPTION
                Makes the dispatching to give a number to the current symptom (if there are no
        selected symptoms) or to give a number to the selected symptoms

  RETURN


  EXAMPLE

  SEE ALSO

*****************************************************************************/

void GetNumberSymptom (num )
int num;
{
VBOEN_OBJECT     *_ptr, _spt1, *_object;
int                 _end, _i, _x, _j, _max, _temp_max;


        /*********************************************************************/
        /* On doit regarder si le symptome courant est un symptome selectione */
        /*                                                                   */
        /* Si il n'est pas selectionne :   on attribue un numero ce symptome          */
        /* Si il est selectionn,       :   on attribue un numeo aux spts. Selectionnes*/
        /*********************************************************************/
```

```
_spt1.Type = OBJECT_VBOEN_SYMPTOM;
_spt1.Id   = (long) WinViewBoen->CurrentSpt;
_object    = (VBOEN_OBJECT *) NULL;
_object    = GetViewBoenObjectInList ( WinViewBoen->Select, &_spt1 );
if ( !_object || WinViewBoen->Select == NULL) {

        /*************************************************************************/
        /* On attribue un numero au current car il n'est pas dans les selectionnes */
        /*************************************************************************/

        GetNumberInMemory (WinViewBoen->SCNumber, WinViewBoen->CurrentSpt,num);

        /*************************************************************************/
        /*  REFRESH de la window */
        /*************************************************************************/

    /*  FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_DISPLAY );
      DisplayViewBoenWindow ( WinViewBoen->Win, VBOEN_DUMMY_REFRESH );*/

}
else {


        /***********************************************/
        /* On compte le nombre de symptomes a attribuer */
        /***********************************************/
        _i   =  1;
        _ptr =  WinViewBoen->Select;
        _max = -1;
        _j   =  0;
        while ( !_j ) {
                        if ( (int)_ptr->Id > _max ) _max = (int) _ptr->Id;
                        if ( _ptr->Next != NULL )   {
                                                        _ptr = _ptr -> Next;
                                                       _i++;
                                                    }
                        else _j = 1;
                      }


        _ptr = WinViewBoen->Select;

        _temp_max =  _max;
        _end      =  0;
        while ( _end < _i ) {
GetNumberInMemory (WinViewBoen->SCNumber,_temp_max,num);
```

```
        _max = _temp_max;
        _end ++;

        _ptr      =  WinViewBoen->Select;
        _temp_max = -1;

        _x = 0;
        while ( !_x ) {
                if ( ((int)_ptr->Id > _temp_max) && ((int)_ptr->Id < _max) )
                        _temp_max = (int)_ptr->Id;

                if (_ptr->Next != NULL) _ptr =  _ptr->Next;
                else                    _x = 1;
        }
               }
               /*****************************************/
               /* On doit liberer les objects selectionnes */
               /*****************************************/
              /*  FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_NO_DISPLAY );
               DisplayViewBoenWindow ( WinViewBoen->Win, VBOEN_GO_TO_FIRST );*/
       }
}



/*****************************************************************************

  NAME
       DeleteNumberInMemory : Deletes a symptom number in memory.

  SYNTAX
void
       DeleteNumberInMemory ( SCNumber, SptNumber );

  PARAMETERS
       int SCNumber                I Clipboard number where to delete spt
       int SptNumber               I Number of the symptom to delete.

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
                Deletes number of ONE symptom in memory

  RETURN


  EXAMPLE
```

```
  SEE ALSO

*****************************************************************************/

void
DeleteNumberInMemory ( _SCNumber, _SptNumber)
   int _SCNumber;
   int _SptNumber;

{
SYMPTOM_IN_MEMORY  *_spt;
int                _i, num;
unsigned char c1;

       if ((_SCNumber==4)|| (SC[_SCNumber].First == NULL) ) {
      /*xvt_dm_post_note("%d",_SCNumber);
      xvt_dm_post_note( GetMessage ( FileMessage, 109, Option.General.Lang, &RecMessage), _SptNumber
);*/
               return;
   }


   _i   = 0;
   _spt = SC[_SCNumber].First;
   while ( (_i < _SptNumber) &&
           (_spt != (SYMPTOM_IN_MEMORY *)NULL) ) {
            _spt = _spt->Next;
            _i++;
   }
   if (_i==_SptNumber)
   {
        c1 = _spt->Qualif;
        for (num =1; c1 !=0; c1 >>=1)
        {
         if (c1 & 1)
           RectButton[num-1].Button = ON;
           DisplayVBoenButton (num-1);
           num++;
         }

        c1 = _spt->Degrees;
        for (num =1; c1 !=0; c1>>=1)
         {
          if (c1 & 1)
           RectButton[num+7].Button = ON;
           DisplayVBoenButton (num+7);
           num++;
          }

        SptInMemory=_spt;
        SptInMemory->Qualif =0;
```

```
        SptInMemory->Degrees=0;

    }

}

/*****************************************************************************

  NAME
               DeleteNumberSymptom : Deletes number of the current or the selected symptoms

  SYNTAX
               void
               DeleteNumberSymptom ( );

  PARAMETERS

  INCLUDE
               xvt.h
               radar.h

  DESCRIPTION
               Makes the dispatching to delete number of the current symptom (if there are no
        selected symptoms) or to delete number of the selected symptoms

  RETURN


  EXAMPLE

  SEE ALSO

*****************************************************************************/

void DeleteNumberSymptom ( )
{
VBOEN_OBJECT     *_ptr, _spt1, *_object;
int                 _end, _i, _x, _j, _max, _temp_max;


       /*********************************************************************/
       /* On doit regarder si le symptome courant est un symptome selectione */
       /*                                                                   */
       /* Si il n'est pas selectionne :   on attribue un numero ce symptome          */
       /* Si il est selectionn,       :   on attribue un numeo aux spts. Selectionnes*/
       /*********************************************************************/
```

```
_spt1.Type = OBJECT_VBOEN_SYMPTOM;
_spt1.Id   = (long) WinViewBoen->CurrentSpt;
_object    = (VBOEN_OBJECT *) NULL;
_object    = GetViewBoenObjectInList ( WinViewBoen->Select, &_spt1 );

if ( !_object || WinViewBoen->Select == NULL) {

        /*************************************************************************/
        /* On attribue un numero au current car il n'est pas dans les selectionnes */
        /*************************************************************************/

        DeleteNumberInMemory (WinViewBoen->SCNumber,WinViewBoen->CurrentSpt);

        /*************************************************************************/
        /*  REFRESH de la window */
        /*************************************************************************/

    /*  FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_DISPLAY );
     DisplayViewBoenWindow ( WinViewBoen->Win, VBOEN_REFRESH  );*/

}
else {


        /***********************************************/
        /* On compte le nombre de symptomes a attribuer */
        /***********************************************/
        _i   =  1;
        _ptr =  WinViewBoen->Select;
        _max = -1;
        _j   =  0;
        while ( !_j ) {
                        if ( (int)_ptr->Id > _max ) _max = (int) _ptr->Id;
                        if ( _ptr->Next != NULL )   {
                                                        _ptr = _ptr -> Next;
                                                      _i++;
                                                }
                        else _j = 1;
                    }


        _ptr = WinViewBoen->Select;

        _temp_max =  _max;
        _end      =  0;
        while ( _end < _i ) {

DeleteNumberInMemory (WinViewBoen->SCNumber,_temp_max);
```

```
        _max = _temp_max;
        _end ++;

        _ptr      =  WinViewBoen->Select;
        _temp_max = -1;

        _x = 0;
        while ( !_x ) {
                if ( ((int)_ptr->Id > _temp_max) && ((int)_ptr->Id < _max) )
                        _temp_max = (int)_ptr->Id;

                if (_ptr->Next != NULL) _ptr =  _ptr->Next;
                else                    _x = 1;
        }
               }
               /*****************************************/
               /* On doit liberer les objects selectionnes */
               /*****************************************/
              /* FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_NO_DISPLAY );
               DisplayViewBoenWindow ( WinViewBoen->Win, VBOEN_GO_TO_FIRST );*/
       }
}


/*************************************************************************/
/*****************************************************************************

  NAME
       DeleteCurrentNumberInMemory : Deletes a symptom in memory.

  SYNTAX
       Void
DeleteCurrentNumberInMemory ( _SCNumber, _SptNumber, num)

  PARAMETERS
   int num;
   int _SCNumber;
   int _SptNumber;

  INCLUDE
       xvt.h
       radar.h

  DESCRIPTION
               Deletes all number to ONE symptom in memory

  RETURN
```

```
  EXAMPLE

  SEE ALSO

*****************************************************************************/

void
DeleteCurrentNumberInMemory ( _SCNumber, _SptNumber, num)
   int num;
   int _SCNumber;
   int _SptNumber;

{
unsigned char c2;
SYMPTOM_IN_MEMORY  *_spt;


int                _i;

       if ((_SCNumber==4)|| (SC[_SCNumber].First == NULL) ) {
      /*xvt_dm_post_note("%d",_SCNumber);
      xvt_dm_post_note( GetMessage ( FileMessage, 109, Option.General.Lang, &RecMessage), _SptNumber
);*/
               return;
       }


   _i    = 0;

    RectButton[num-1].Button = ON;
    DisplayVboenButton(num-1);

   _spt = SC[_SCNumber].First;
   while ( (_i < _SptNumber) &&
           (_spt != (SYMPTOM_IN_MEMORY *)NULL) ) {
            _spt = _spt->Next;
            _i++;
   }
   if (_i==_SptNumber)
   {
        if (num >0 && num<9)
        {
         c2=1;
         c2 = c2 << (num-1);
        if(_spt->Qualif & c2)
               {
               _spt->Qualif= _spt->Qualif & ~c2;
               SptInMemory =_spt;
               }
        }
        if(num >8 && num<17)
```

```
        {
          c2=1;
          c2 = c2 << (num-9);
          if(_spt->Degrees & c2)
                {
                _spt->Degrees = _spt->Degrees & ~c2;
                SptInMemory=_spt;
                }

        }

     }


}

/*****************************************************************************

  NAME
                DeleteCurrentNumberSymptom : Deletes all number to the current or the selected symptoms

  SYNTAX
void DeleteCurrentNumberSymptom (num )

  PARAMETERS
int num;

  INCLUDE
                xvt.h
                radar.h

  DESCRIPTION
                Makes the dispatching to delete all number to the current symptom (if there are no
        selected symptoms) or to delete all number to the selected symptoms

  RETURN


  EXAMPLE

  SEE ALSO

*****************************************************************************/

void DeleteCurrentNumberSymptom (num )
int num;
{
VBOEN_OBJECT      *_ptr, _spt1, *_object;
int                  _end, _i, _x, _j, _max, _temp_max;
```

```
/*********************************************************************/
/* On doit regarder si le symptome courant est un symptome selectione */
/*                                                                    */
/* Si il n'est pas selectionne :   on attribue un numero ce symptome          */
/* Si il est selectionn,       :   on attribue un numeo aux spts. Selectionnes*/
/*********************************************************************/


_spt1.Type = OBJECT_VBOEN_SYMPTOM;
_spt1.Id   = (long) WinViewBoen->CurrentSpt;
_object    = (VBOEN_OBJECT *) NULL;
_object    = GetViewBoenObjectInList ( WinViewBoen->Select, &_spt1 );

if ( !_object || WinViewBoen->Select == NULL) {

        /*********************************************************************/
        /* On attribue un numero au current car il n'est pas dans les selectionnes */
        /*********************************************************************/

        DeleteCurrentNumberInMemory (WinViewBoen->SCNumber,WinViewBoen->CurrentSpt,num);

        /*********************************************************************/
        /*  REFRESH de la window */
        /*********************************************************************/

     /* FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_DISPLAY );
      DisplayViewBoenWindow ( WinViewBoen->Win, VBOEN_REFRESH );*/

}
else {


        /*********************************************/
        /* On compte le nombre de symptomes a attribuer */
        /*********************************************/
        _i   =  1;
        _ptr =  WinViewBoen->Select;
        _max = -1;
        _j   =  0;
        while ( !_j ) {
                        if ( (int)_ptr->Id > _max ) _max = (int) _ptr->Id;
                        if ( _ptr->Next != NULL )   {
                                                        _ptr = _ptr -> Next;
                                                       _i++;
                                                   }
                        else _j = 1;
                    }
```

```
                _ptr = WinViewBoen->Select;

                _temp_max =  _max;
                _end      =  0;
                while ( _end < _i ) {

        DeleteCurrentNumberInMemory (WinViewBoen->SCNumber,_temp_max,num);



         _max = _temp_max;
         _end ++;

         _ptr      =  WinViewBoen->Select;
         _temp_max = -1;

         _x = 0;
         while ( !_x ) {
                if ( ((int)_ptr->Id > _temp_max) && ((int)_ptr->Id < _max) )
                        _temp_max = (int)_ptr->Id;

                if (_ptr->Next != NULL) _ptr =  _ptr->Next;
                else                    _x = 1;
         }
                }
                /*****************************************/
                /* On doit liberer les objects selectionnes */
                /*****************************************/
               /* FreeViewBoenObject ( VBOEN_KEEP_SELECT, VBOEN_KEEP_NO_DISPLAY );
                DisplayViewBoenWindow ( WinViewBoen->Win, VBOEN_GO_TO_FIRST );*/
        }
}
```

A.3.3.1 FONCTION X_NSYMP

```
/*
        This file was generated by XVT-Design 3.03, a product of:

                XVT Software Inc.
                4900 Pearl East Circle
                Boulder, CO USA 80301
                303-443-4223, fax 303-443-0969

        Generated on Sun Mar 23 12:31:39 1997
*/

#include "xvt.h"
#include "xvtcm.h"
#include "x_radar.h"
#include "radar.h"
```

```
/*
        Information about the window
*/
#define WIN_RES_ID W_NSYMP
#define WIN_FLAGS 0x802L
#define WIN_CLASS ""
#define WIN_BORDER W_DOC

/*
        Handler for window W_NSYMP ("Number Symptoms")
*/
long XVT_CALLCONV1
#if XVT_CC_PROTO
W_NSYMP_eh XVT_CALLCONV2 (WINDOW xdWindow, EVENT *xdEvent)
#else
W_NSYMP_eh XVT_CALLCONV2 (xdWindow, xdEvent)
WINDOW xdWindow;
EVENT *xdEvent;
#endif
{
        short xdControlId = xdEvent->v.ctl.id;
                static WIN_MSG WinMsgNumber [] = {
                        W_NSYMP,                        -1,   '\0',
                        W_NSYMP_OK,                    855,   '\0',
                        W_NSYMP_CANCEL,                856,   '\0',
                        0,                                0,      '\0' };


        switch (xdEvent->type) {
        case E_CREATE:
                /*
                        Window has been created; first event sent to newly-created
                        window.
                */
                {
                 LoadMessagesForWindow( xdWindow,(WIN_MSG *) WinMsgNumber);
                }
                break;
        case E_DESTROY:
                /*
                        Window has been closed; last event sent to window.
                */
                xdRemoveHelpAssoc( xdWindow );
                {
                }
                break;
        case E_FOCUS:
                {
                /*
                        Window has lost or gained focus.
```

```
        */
        if (xdEvent->v.active)  {
                /*
                        Window has gained focus
                */
        } else {
                /*
                        Window has lost focus
                */
        }
        }
        break;
case E_SIZE:
        /*
                Size of window has been set or changed; sent when window is
                created or subsequently resized by user or  via xvt_vobj_move.
        */
        {
        }
        break;
case E_UPDATE:
        /*
                Window requires updating.
        */
        {
        UpdateNumberWindow(xdWindow);
        }
        break;
case E_CLOSE:
        /*
                Request to close window; user operated "close" menu item on
                window system menu, or operated "close" control on window
                frame. Not sent if Close on File menu is issued. Window not
                closed unless xvt_vobj_destroy is called.
        */
        {
        xvt_vobj_destroy(xdWindow);
        }
        break;
case E_CHAR:
        /*
          .     Character typed.
        */
        {
        }
        break;
case E_MOUSE_UP:
        /*
                Mouse was released
        */
```

```
        {
        }
        break;
case E_MOUSE_DOWN:
        /*
                Mouse was pressed
        */
        {
        }
        break;
case E_MOUSE_DBL:
        /*
                Mouse was double clicked
        */
        {
        }
        break;
case E_MOUSE_MOVE:
        /*
                Mouse was moved
        */
        {
        }
        break;
case E_HSCROLL:
        {
        /*
                Horizontal scrollbar on frame was operated
        */
        switch (xdEvent->v.scroll.what) {
        case SC_LINE_UP:
                break;
        case SC_LINE_DOWN:
                break;
        case SC_PAGE_UP:
                break;
        case SC_PAGE_DOWN:
                break;
        case SC_THUMB:
                break;
        case SC_THUMBTRACK:
                break;
        default:
                break;
        }
        }
        break;
case E_VSCROLL:
        {
        /*
```

```
                Vertical scrollbar on frame was operated
        */
        switch (xdEvent->v.scroll.what) {
        case SC_LINE_UP:
                break;
        case SC_LINE_DOWN:
                break;
        case SC_PAGE_UP:
                break;
        case SC_PAGE_DOWN:
                break;
        case SC_THUMB:
                break;
        case SC_THUMBTRACK:
                break;
        default:
                break;
        }
        }
        break;
case E_COMMAND:
        /*
                User issued command on window menu bar (menu bar at top of
                screen for Mac/CH).
        */
        {
        /*
                No menubar was associated with this window
        */
        }
        break;
case E_CONTROL:
        /*
                User operated control in window.
        */
        {
          char _buf [4];
          int  _i;

        switch(xdControlId) {
        case W_NSYMP_EDIT:                      {
                /*
                        Edit control was operated.
                */
                if (xdEvent->v.ctl.ci.v.edit.focus_change) {
                        if (xdEvent->v.ctl.ci.v.edit.active) {
                                /*
                                        focus has entered the control
                                */
                        } else {
```

```
                                /*
                                        focus has left the control
                                */
                        }
                } else {
                        /*
                                Contents of control were changed
                        */
                }
                }
                break;
        case W_NSYMP_OK: /* "OK" */
                {
                xvt_vobj_get_title(xvt_win_get_ctl(xdWindow,W_NSYMP_EDIT), (char *) _buf, 4);
                _i=atoi(_buf);
                GetNumberSymptom(_i);
                xvt_vobj_destroy(xdWindow);
                }
                break;
        case W_NSYMP_CANCEL: /* "Cancel" */
                {
                xvt_vobj_destroy(xdWindow);
                }
                break;
        default:
                break;
        }
        }
        break;
case E_FONT:
        /*
                User issued font command on window menu bar (menu bar at top of
                screen for Mac/CH).
        */
        {
        }
        break;
case E_TIMER:
        /*
                Timer associated with window went off.
        */
        {
        }
        break;
case E_USER:
        /*
                Application initiated.
        */
        {
        switch (xdEvent->v.user.id) {
```

```
                case -1:
                default:
                        break;
                }
                }
                break;
        default:
                break;
        }
        xvt_tx_process_event(xdWindow, xdEvent);
        return 0L;
}
```

A.3.3.2 FONCTION X_DELNUM

```
/*
        This file was generated by XVT-Design 3.03, a product of:

                XVT Software Inc.
                4900 Pearl East Circle
                Boulder, CO USA 80301
                303-443-4223, fax 303-443-0969

        Generated on Mon Mar 24 12:36:20 1997
*/

#include "xvt.h"
#include "xvtcm.h"
#include "x_radar.h"
#include "radar.h"


/*
        Information about the window
*/
#define WIN_RES_ID W_DNUM
#define WIN_FLAGS 0x802L
#define WIN_CLASS ""
#define WIN_BORDER W_DOC

/*
        Handler for window W_DNUM ("Number Symptom")
*/
long XVT_CALLCONV1
#if XVT_CC_PROTO
W_DNUM_eh XVT_CALLCONV2 (WINDOW xdWindow, EVENT *xdEvent)
#else
W_DNUM_eh XVT_CALLCONV2 (xdWindow, xdEvent)
WINDOW xdWindow;
EVENT *xdEvent;
```

```
#endif
{
        short xdControlId = xdEvent->v.ctl.id;

                static WIN_MSG WinMsgNum [] = {
                        W_DNUM,                         -1,    '\0',
                        W_DNUM_OK,                     855,    '\0',
                        W_DNUM_CANCEL,                 856,    '\0',
                        0,                               0,      '\0' };

        switch (xdEvent->type) {
        case E_CREATE:
                /*
                        Window has been created; first event sent to newly-created
                        window.
                */
                {
                 LoadMessagesForWindow(xdWindow, (WIN_MSG *)WinMsgNum);
                }
                break;
        case E_DESTROY:
                /*
                        Window has been closed; last event sent to window.
                */
                xdRemoveHelpAssoc( xdWindow );
                {
                }
                break;
        case E_FOCUS:
                {
                /*
                        Window has lost or gained focus.
                */
                if (xdEvent->v.active)  {
                        /*
                                Window has gained focus
                        */
                } else {
                        /*
                                Window has lost focus
                        */
                }
                }
                break;
        case E_SIZE:
                /*
                        Size of window has been set or changed; sent when window is
                        created or subsequently resized by user or via xvt_vobj_move.
                */
                {
```

```
        }
        break;
case E_UPDATE:
        /*
                Window requires updating.
        */
        {
        UpdateNumberWindow(xdWindow);
        }
        break;
case E_CLOSE:
        /*
                Request to close window; user operated "close" menu item on
                window system menu, or operated "close" control on window
                frame. Not sent if Close on File menu is issued. Window not
                closed unless xvt_vobj_destroy is called.
        */
        {
        xvt_vobj_destroy(xdWindow);
        }
        break;
case E_CHAR:
        /*
                Character typed.
        */
        {
        }
        break;
case E_MOUSE_UP:
        /*
                Mouse was released
        */
        {
        }
        break;
case E_MOUSE_DOWN:
        /*
                Mouse was pressed
        */
        {
        }
        break;
case E_MOUSE_DBL:
        /*
                Mouse was double clicked
        */
        {
        }
        break;
case E_MOUSE_MOVE:
```

```
        /*
                Mouse was moved
        */
        {
        }
        break;
case E_HSCROLL:
        {
        /*
                Horizontal scrollbar on frame was operated
        */
        switch (xdEvent->v.scroll.what) {
        case SC_LINE_UP:
                break;
        case SC_LINE_DOWN:
                break;
        case SC_PAGE_UP:
                break;
        case SC_PAGE_DOWN:
                break;
        case SC_THUMB:
                break;
        case SC_THUMBTRACK:
                break;
        default:
                break;
        }
        }
        break;
case E_VSCROLL:
        {
        /*
                Vertical scrollbar on frame was operated
        */
        switch (xdEvent->v.scroll.what) {
        case SC_LINE_UP:
                break;
        case SC_LINE_DOWN:
                break;
        case SC_PAGE_UP:
                break;
        case SC_PAGE_DOWN:
                break;
        case SC_THUMB:
                break;
        case SC_THUMBTRACK:
                break;
        default:
                break;
        }
```

```
        }
        break;
case E_COMMAND:
        /*
                User issued command on window menu bar (menu bar at top of
                screen for Mac/CH).
        */
        {
        /*
                No menubar was associated with this window
        */
        }
        break;
case E_CONTROL:
        /*
                User operated control in window.
        */
        {
           char _buf[4];
           int _i;
        switch(xdControlId) {
        case W_DNUM_EDIT:                       {
                /*
                        Edit control was operated.
                */
                if (xdEvent->v.ctl.ci.v.edit.focus_change) {
                        if (xdEvent->v.ctl.ci.v.edit.active) {
                                /*
                                        focus has entered the control
                                */
                        } else {
                                /*
                                        focus has left the control
                                */
                        }
                } else {
                        /*
                                Contents of control were changed
                        */
                }
                }
                break;
        case W_DNUM_CANCEL: /* "Cancel" */
                {
                xvt_vobj_destroy(xdWindow);

                }
                break;
        case W_DNUM_OK: /* "OK" */
                {
```

```
                xvt_vobj_get_title(xvt_win_get_ctl(xdWindow,W_DNUM_EDIT), (char *) _buf, 4);
                _i=atoi(_buf);
                DeleteCurrentNumberSymptom(_i);
                xvt_vobj_get_title(xvt_win_get_ctl(xdWindow,W_DNUM_EDIT), (char *) _buf, 4);
                _i=atoi(_buf);
                DeleteCurrentNumberSymptom(_i);
                xvt_vobj_destroy(xdWindow);
                }
                break;
        default:
                break;
        }
        }
        break;
case E_FONT:
        /*
                User issued font command on window menu bar (menu bar at top of
                screen for Mac/CH).
        */
        {
        }
        break;
case E_TIMER:
        /*
                Timer associated with window went off.
        */
        {
        }
        break;
case E_USER:
        /*
                Application initiated.
        */
        {
        switch (xdEvent->v.user.id) {
        case -1:
        default:
                break;
        }
        }
        break;
default:
        break;
}
xvt_tx_process_event(xdWindow, xdEvent);
return 0L;
}
```

A.3.4 SOURCE CLINICAL

```
/*****************************************************************************

        SOURCE FILE
                CLINIQUE.C

        MODULE
                View Clinique Symptoms window

        AUTHOR
              Diagne Papa Moussa            26/09/96

        DESCRIPTION
                Contain all functions for the management of the
                View Clinique Symptoms window



/*---- INCLUDE SECTION ----------------------------------------------------*/

#include "xvt.h"
#include "x_radar.h"
#include "radar.h"
#include "math.h"
/*---- DEFINE SECTION -----------------------------------------------------*/


/*---- VARIABLE SECTION ---------------------------------------------------*/

/*---- LOW LEVEL FUNCTIONS PROTOTYPES -------------------------------------*/

/*---- HIGH FUNCTION SECTION ----------------------------------------------*/


/*****************************************************************/
Cette fonction crée et initialise la fenêtre clinical case
/*****************************************************************/
void
InitializeViewCliniqueWindow ( )
{
WINDOW  _clin;
WIN_DEF *_windef;

        xvt_win_set_cursor ( TASK_WIN, CURSOR_WAIT );
        DbVirtualClose ( 2 );

      if ( ConsWindow && !WinRep ) {
         xvt_dm_post_note( GetMessage ( FileMessage, 4036, Option.General.Lang, &RecMessage) );
```

```
        return;
      }

      if ( !ConsWindow && WinRep ) {
        xvt_dm_post_note( GetMessage ( FileMessage, 4035, Option.General.Lang, &RecMessage) );
        return;
      }



       if ( ! ( WinViewClin = (CLIN_WINDOW *)Malloc ( sizeof ( CLIN_WINDOW ) + 1 ) ) ) {
              xvt_dm_post_note( GetMessage ( FileMessage, 104, Option.General.Lang, &RecMessage)  );
              xvt_win_set_cursor ( TASK_WIN, CURSOR_ARROW );
              return;
       }


       WinViewClin->Type           = WINDOW_TYPE_CLIN;
       CalculateSymptomMaladie();
       CalculateSymptomMalade();
       DbVirtualClose ( 2 );

/* Create and calculate window */

        if ( ! ( _windef = xvt_res_get_win_def ( WIN_CCASE ) ) ) {
              xvt_dm_post_note( GetMessage ( FileMessage, 31, Option.General.Lang, &RecMessage ) );
              Free ( (char *)WinViewClin );
              xvt_win_set_cursor ( TASK_WIN, CURSOR_ARROW );
              return;
        }


        if ( !( _clin = xvt_win_create_def ( _windef, TASK_WIN, EM_ALL,
                                        WIN_CCASE_eh, (long)WinViewClin ) ) ) {
              xvt_dm_post_note( GetMessage ( FileMessage, 31, Option.General.Lang, &RecMessage ) );
              Free ( (char *)WinViewClin );
              xvt_win_set_cursor ( TASK_WIN, CURSOR_ARROW );
              return;
        }
        xvt_res_free_win_def ( _windef );

       WinViewClin->Win                = _clin;
       WinViewClin->Focus              = 0;

      UpdatePatientCase();

}
/*********************************************************/
Cette fonction met à jour la fenêtre clinical case
/*********************************************************/
```

```
void   UpdateViewCliniqueWindow( win)
WINDOW win;
{
                short       _h,
                            _v;
                 RCT rct1;
                 CBRUSH brush;
                 CPEN   pen;

                 xvt_dwin_clear ( win, Option.Patient.ColorBg ) ;
                 xvt_dwin_set_fore_color ( win, COLOR_BLACK ) ;
                 DisplayViewMaladieSymptom(win, CLIN_REFRESH_2);
                 DisplayViewMaladeSymptom(win, CLIN_REFRESH_1);


  pen.color=Option.Patient.ColorBg;
  pen.pat=PAT_SOLID;
  pen.style=P_SOLID;
  pen.width=1;
  xvt_dwin_set_cpen ( win, &pen ) ;
  brush.pat=PAT_SOLID;
  brush.color=Option.Patient.ColorBg;
  xvt_dwin_set_cbrush ( win, &brush ) ;
  xvt_rect_set(&rct1,5,2,633,59); /*x=0*/
  xvt_dwin_draw_rect ( win, &rct1 ) ;
  DrawSunkenBox ( win, &rct1 ) ;
  _v = 24;
  WindowChangeResolution ( &_v, WINDOW_VERT, RESOLUTION_UP );


  _h = 10;
  WindowChangeResolution ( &_h, WINDOW_HOR, RESOLUTION_UP );
  xvt_dwin_draw_text ( win, _h, _v, GetMessage ( FileMessage, 4030, Option.General.Lang, &RecMessage ),
-1 );


  _h = 315;
  WindowChangeResolution ( &_h, WINDOW_HOR, RESOLUTION_UP );
  xvt_dwin_draw_text ( win, _h, _v, GetMessage ( FileMessage, 1109, Option.General.Lang, &RecMessage ),
-1 );


  _h = 465;
  WindowChangeResolution ( &_h, WINDOW_HOR, RESOLUTION_UP );
  xvt_dwin_draw_text ( win, _h, _v, GetMessage ( FileMessage, 1011, Option.General.Lang, &RecMessage ),
-1 );


  _v = 50;
  _h = 70;
  WindowChangeResolution ( &_h, WINDOW_HOR, RESOLUTION_UP );
  xvt_dwin_draw_text ( win, _h, _v, GetMessage ( FileMessage, 1000, Option.General.Lang, &RecMessage ),
-1 );


  _h = 240;/*h=245*/
```

```
  WindowChangeResolution ( &_h, WINDOW_HOR, RESOLUTION_UP );
  xvt_dwin_draw_text ( win, _h, _v, GetMessage ( FileMessage, 1001, Option.General.Lang, &RecMessage ),
-1 );



  _h = 426;
  WindowChangeResolution ( &_h, WINDOW_HOR, RESOLUTION_UP );
  xvt_dwin_draw_text ( win, _h, _v, GetMessage ( FileMessage, 4031, Option.General.Lang, &RecMessage ),
-1 );



  xvt_rect_set(&rct1,5,65,633,100);
  xvt_dwin_draw_rect ( win, &rct1 ) ;
  DrawSunkenBox ( win, &rct1 ) ;


  _v = 90;
  _h = 30;
  WindowChangeResolution ( &_h, WINDOW_HOR, RESOLUTION_UP );
  xvt_dwin_draw_text ( win, _h, _v, GetMessage ( FileMessage, 4032, Option.General.Lang, &RecMessage ),
-1 );


  _h = 390;
  WindowChangeResolution ( &_h, WINDOW_HOR, RESOLUTION_UP );
  xvt_dwin_draw_text ( win, _h, _v, GetMessage ( FileMessage, 1110, Option.General.Lang, &RecMessage ),
-1 );


  xvt_rect_set(&rct1,5,105,543,188);
  xvt_dwin_draw_rect ( win, &rct1 ) ;
  DrawSunkenBox ( win, &rct1 ) ;


  _v = 120;
  _h = 25;
  WindowChangeResolution ( &_h, WINDOW_HOR, RESOLUTION_UP );
  xvt_dwin_draw_text ( win, _h, _v, GetMessage ( FileMessage, 1112, Option.General.Lang, &RecMessage ),
-1 );


  xvt_rect_set(&rct1,548,105,633,188);
  xvt_dwin_draw_rect ( win, &rct1 ) ;
  DrawSunkenBox ( win, &rct1 ) ;


  _v = 120;
  _h = 305;
  WindowChangeResolution ( &_h, WINDOW_HOR, RESOLUTION_UP );
  xvt_dwin_draw_text ( win, _h, _v, GetMessage ( FileMessage, 134, Option.General.Lang, &RecMessage ),
-1 );


 }
/*********************************************************/
Cette fonction met à jour les fenêtres give et delete number symptom
/*********************************************************/
void   UpdateNumberWindow( win)
WINDOW win;
```

```
{
                short       _h,
                            _v;
                 RCT rct1;
                 CBRUSH brush;
                 CPEN   pen;

                 xvt_dwin_clear ( win, Option.Patient.ColorBg ) ;
                 xvt_dwin_set_fore_color ( win, COLOR_BLACK ) ;


  pen.color=Option.Patient.ColorBg;
  pen.pat=PAT_SOLID;
  pen.style=P_SOLID;
  pen.width=1;
  xvt_dwin_set_cpen ( win, &pen ) ;
  brush.pat=PAT_SOLID;
  brush.color=Option.Patient.ColorBg;
  xvt_dwin_set_cbrush ( win, &brush ) ;
  xvt_rect_set(&rct1,1,5,167,150);
  xvt_dwin_draw_rect ( win, &rct1 ) ;
  DrawSunkenBox ( win, &rct1 ) ;
  xvt_rect_set(&rct1,168,5,274,150);
  xvt_dwin_draw_rect ( win, &rct1 ) ;
  DrawSunkenBox ( win, &rct1 ) ;

  _v = 70;
  WindowChangeResolution ( &_v, WINDOW_VERT, RESOLUTION_UP );

  _h = 15;
  WindowChangeResolution ( &_h, WINDOW_HOR, RESOLUTION_UP );
  xvt_dwin_draw_text ( win, _h, _v, GetMessage ( FileMessage, 4024, Option.General.Lang, &RecMessage ),
-1 );


  }
/*************************************************************/
Cette fonction met à jour les données du patients
/*************************************************************/
  void UpdatePatientCase()
 {
  char _s1[10];
  char _res[10];
  char  _buf1[20],_buf[80];
  int   i;

  xvt_vobj_set_title(xvt_win_get_ctl(WinViewClin->Win,WIN_CCASE_NOMDOSSIER), SaveCaseData.CaseName);
  xvt_vobj_set_title(xvt_win_get_ctl(WinViewClin->Win,WIN_CCASE_EDIT_NOM), PatRec.Name);
  xvt_vobj_set_title(xvt_win_get_ctl(WinViewClin->Win,WIN_CCASE_EDIT_PRENOM), PatRec.FirstName);
  xvt_vobj_set_title(xvt_win_get_ctl(WinViewClin->Win,WIN_CCASE_EDIT_DIAGNOS), DefDiagRec.Name);
```

```
  DateLongToChar ( Option.General.Lang, PatRec.Birth, Option.General.FormatDate, _s1 ) ;
  xvt_vobj_set_title ( xvt_win_get_ctl ( WinViewClin->Win, WIN_CCASE_EDIT_NAISSANCE ), _s1 ) ;

  DateLongToChar ( Option.General.Lang, ConsRec.Date, Option.General.FormatDate, _s1 ) ;
  xvt_vobj_set_title ( xvt_win_get_ctl ( WinViewClin->Win, WIN_CCASE_EDIT_DATECONS ), _s1 ) ;

  strcpy ( _res, GetMessage ( FileMessage, 1088 + ConsRec.Result,
                              Option.General.Lang, &RecMessage ) );
  xvt_vobj_set_title(xvt_win_get_ctl(WinViewClin->Win,WIN_CCASE_EDIT_RESULT),_res);

  if (PatRec.Sex == PAT_SEX_MALE)
  xdCheckRadioButton(WinViewClin->Win, WIN_CCASE_MALE, WIN_CCASE_MALE, WIN_CCASE_FEMELLE);

  if(PatRec.Sex == PAT_SEX_FEMALE)
  xdCheckRadioButton(WinViewClin->Win, WIN_CCASE_FEMELLE, WIN_CCASE_MALE, WIN_CCASE_FEMELLE);

  DisplayTextEdit(xvt_win_get_tx(WinViewClin->Win,WIN_CCASE_REMEDE),ConsRec.Comment);

 /* sprintf(_buf,"");
  sprintf(_buf,"%s  ",PremRec.Product);
  strcat( _buf, PremRec.Importance);
  DisplayTextEdit(xvt_win_get_tx(WinViewClin->Win,WIN_CCASE_REMARQUE),_buf);*/

  PremRec.ConsId = ConsRec.Id;
  HrsFindFullPrem ( PremFile, &PremRec, PREM_IDX_CONS, DB_GTEQ ) ;

     sprintf(_buf,"%s  ",PremRec.Product);
     sprintf( _buf1,"%s\n", PremRec.Importance);
     strcat(_buf, _buf1);
     HrsFindFullPrem ( PremFile, &PremRec, PREM_IDX_CONS, DB_NEXT ) ;

  while( PremRec.ConsId == ConsRec.Id && RadarError == DB_OK )
   {
     sprintf(_buf1,"%s  ",PremRec.Product);
     strcat(_buf, _buf1);
     sprintf( _buf1,"%s\n", PremRec.Importance);
     strcat(_buf, _buf1);
     HrsFindFullPrem ( PremFile, &PremRec, PREM_IDX_CONS, DB_NEXT ) ;
   }
   DisplayTextEdit(xvt_win_get_tx(WinViewClin->Win,WIN_CCASE_REMARQUE),_buf);
}

/*********************************************************/
Cette fonction affiche les symptômes de la maladie du cas
/*********************************************************/

void DisplayViewMaladieSymptom(win,mode)

     WINDOW        win;
     int           mode;
```

```
{

  SYMPTOM_IN_MEMORY *_Spt;

            RCT                _rct;
            PNT                _from,
                               _from2,
                                 _to;
           short               _leading,
                               _ascent,
                              _descent;
           int                 num,
                               _i,
                               _st,
                               _j,
                             _width;
            char              _buf[50],
                             _buf1[4];
            unsigned char     c1=0;

    switch ( mode ) {


           case CLIN_REFRESH_2:
                         WinViewClin->FirstSpt_2 = 0;
                         WinViewClin->LastSpt_2 = 0;
                         WinViewClin->CurrentSpt_2 = 0;

                  break;

           case CLIN_LINE_DOWN_2:


                  if (WinViewClin->FirstSpt_2 < (int) SC[4].NbSpt-1)
                    {
                     WinViewClin->FirstSpt_2++;
                     WinViewClin->LastSpt_2++;
                     }
                  else  break;
                  break;

           case CLIN_PAGE_DOWN_2:


                  if (WinViewClin->LastSpt_2 < (int) SC[4].NbSpt)
                    {
                     WinViewClin->FirstSpt_2= WinViewClin->LastSpt_2;
                     }
                  else  break;
                  break;
```

```
                case CLIN_LINE_UP_2:

                                WinViewClin->CurrentSpt_2 = WinViewClin->LastSpt_2;
                                WinViewClin->FirstSpt_2 --;
                                if (WinViewClin->FirstSpt_2<0)
                                  {
                                   WinViewClin->FirstSpt_2 =0;
                                   WinViewClin->LastSpt_2 = WinViewClin->C urrentSpt_2;
                                  }
                               else WinViewClin->LastSpt_2--;
                        break;

                case CLIN_PAGE_UP_2:

                                WinViewClin->CurrentSpt_2 = WinViewClin->LastSpt_2;
                                WinViewClin->FirstSpt_2-= 4;

                                if (WinViewClin->FirstSpt_2<0)
                                  {
                                   WinViewClin->FirstSpt_2 = 0;
                                   WinViewClin->LastSpt_2 = WinViewClin->CurrentSpt_2;
                                  }
                                else WinViewClin->LastSpt_2-=4;
                        break;

                case CLIN_DUMMY_REFRESH_2:
                        break;

        }

              _st =0;
              _i=WinViewClin->FirstSpt_2;
                 xvt_vobj_get_outer_rect(xvt_win_get_ctl(win,WIN_CCASE_VSCROLL_2),&_rct);
                 xvt_rect_set( &WinViewClin->MaladieRect,4,_rct.top,_rct.left,_rct.bottom);
                 WindowDrawRect(win,&WinViewClin->MaladieRect , (COLOR)
xvt_vobj_get_attr(win,ATTR_BACK_COLOR));
                 WindowDrawEmptyRect(win,&WinViewClin->MaladieRect , COLOR_BLACK);
                        strcpy (_buf,  GetMessage (FileMessage, 4015, Option.General.Lang, &RecMessage)
);
                        _width = GetWindowObjectDim ( win, OBJECT_GENERAL_SYMPTOM_LANG_1, _buf,
                                        &_leading, &_ascent, &_descent );
                        _from.v = WinViewClin->MaladieRect.top+4;
                        _from.h = WinViewClin->MaladieRect.left+150 ;
                        _from2.v  = _from.v;
                        _from2.h  = _from.h;
                        DisplayWindowObject ( win, &_from, &_to, OBJECT_SYMPTOM_CAUS_LANG_1, _buf,
                                        "", ATTRIBUTE_NORMAL );
                        _from.v = _from2.v +14;
                        _from.h = WinViewClin->MaladieRect.left+4 ;
```

```
if ( (_Spt = SC[4].First) != NULL)
 {
  for(_j=0;_j<_i;_j++)_Spt=_Spt->Next;

    do {
          c1= _Spt->Qualif;
          for(num=1;c1!=0;c1>>=1) {
           if(c1&1)
            {
             sprintf (_buf1,"%d",num);
             strcat (_buf1,".") ;
             }
             num++;
            }
          _width = GetWindowObjectDim ( win, OBJECT_GENERAL_SYMPTOM_LANG_1, _buf1,
                                        &_leading, &_ascent, &_descent );
           DisplayWindowObject ( win, &_from, &_to,OBJECT_GENERAL_SYMPTOM_LANG_1, _buf1,
                                       "", ATTRIBUTE_NORMAL );

           _from2.h = _from.h;
           _from.h = _from2.h +2+_width;

       _width = GetWindowObjectDim ( win, OBJECT_GENERAL_SYMPTOM_LANG_1,
                                        _Spt->SptTextLg1,
                                         &_leading, &_ascent, &_descent );
         DisplayWindowObject ( win, &_from, &_to,OBJECT_GENERAL_SYMPTOM_LANG_1,
                                       _Spt->SptTextLg1,
                                       "", ATTRIBUTE_NORMAL );
          _st++;
          _from2.v = _from.v;
          _from.v = _from2.v +14;
          _from.h = WinViewClin->MaladieRect.left+4 ;

      _Spt = _Spt->Next;
     }
    while((_Spt!=NULL)&&(_st<4));

    if (mode == CLIN_REFRESH_2)
    WinViewClin->LastSpt_2 = _st;

    if ( mode != CLIN_REFRESH_2 ) {
        if ( WinViewClin->FirstSpt_2 == 0 ) {
               SetWindowScrollBar ( xvt_win_get_ctl ( win, WIN_CCASE_VSCROLL_2 ),
                                       HVSCROLL, 1L, (long)SC[4].NbSpt,
                                       (long) 1 );
        }
        else {
```

```
                        SetWindowScrollBar ( xvt_win_get_ctl ( win, WIN_CCASE_VSCROLL_2 ),
                                             HVSCROLL, 1L, (long)SC[4].NbSpt,
                                             (long)( WinViewClin->FirstSpt_2 ) );
                    }
                }
            }
    }

/*********************************************************/
Cette fonction affiche les symptômes du malade
/*********************************************************/
void DisplayViewMaladeSymptom(win,mode)

    WINDOW      win;
    int         mode;
    {

  SYMPTOM_IN_MEMORY *_Spt;

            RCT             _rct;
            PNT             _from,
                            _from2,
                              _to;
            short           _leading,
                            _ascent,
                            _descent;
            int             num,
                            _i,
                            _st,
                            _j,
                            _width;
            char            _buf[50],
                            _buf1[4];
            unsigned char   c1=0;

      switch ( mode ) {


            case CLIN_REFRESH_1:
                        WinViewClin->FirstSpt_1 = 0 ;
                        WinViewClin->LastSpt_1 = 0;
                        WinViewClin->CurrentSpt_1 = 0;

                    break;

            case CLIN_LINE_DOWN_1:


                    if (WinViewClin->FirstSpt_1 < (int) SC[5].NbSpt-1)
                      {
```

```
                WinViewClin->FirstSpt_1++;
                WinViewClin->LastSpt_1++;
                }
            else  break;
            break;

        case CLIN_PAGE_DOWN_1:


            if (WinViewClin->LastSpt_1 < (int) SC[5].NbSpt)
              {
               WinViewClin->FirstSpt_1= WinViewClin->LastSpt_1;
               }
            else  break;
            break;


        case CLIN_LINE_UP_1:

                    WinViewClin->CurrentSpt_1 = WinViewClin->LastSpt_1;
                    WinViewClin->FirstSpt_1 --;
                    if (WinViewClin->FirstSpt_1<0)
                      {
                       WinViewClin->FirstSpt_1 =0;
                       WinViewClin->LastSpt_1 = WinViewClin->CurrentSpt_1;
                      }
                   else WinViewClin->LastSpt_1--;
            break;

        case CLIN_PAGE_UP_1:

                    WinViewClin->CurrentSpt_1 = WinViewClin->LastSpt_1;
                    WinViewClin->FirstSpt_1-= 4;

                    if (WinViewClin->FirstSpt_1<0)
                      {
                       WinViewClin->FirstSpt_1 = 0;
                       WinViewClin->LastSpt_1 = WinViewCli n->CurrentSpt_1;
                      }
                    else WinViewClin->LastSpt_1-=4;
            break;

        case CLIN_DUMMY_REFRESH_1:
            break;

}

    _st =0;
    _i=WinViewClin->FirstSpt_1;
      xvt_vobj_get_outer_rect(xvt_win_get_ctl(win,WIN_CCASE_VSCROLL_1),&_rct);
```

```
xvt_rect_set( &WinViewClin->MaladeRect,4,_rct.top,_rct.left,_rct.bottom);
WindowDrawRect(win,&WinViewClin->MaladeRect , (COLOR)
                xvt_vobj_get_attr(win,ATTR_BACK_COLOR));
WindowDrawEmptyRect(win,&WinViewClin->MaladeRect , COLOR_BLACK);
strcpy (_buf,  GetMessage (FileMessage, 4016, Option.General.Lang, &RecMessage) );
        _width = GetWindowObjectDim ( win, OBJECT_GENERAL_SYMPTOM_LANG_1, _buf,
                        &_leading, &_ascent, &_descent );
        _from.v = WinViewClin->MaladeRect.top+4;
        _from.h = WinViewClin->MaladeRect.left+150 ;
        _from2.v  = _from.v;
        _from2.h  = _from.h;
        DisplayWindowObject ( win, &_from, &_to, OBJECT_SYMPTOM_CAUS_LANG_1, _buf,
                        "", ATTRIBUTE_NORMAL );
        _from.v = _from2.v +14;
        _from.h = WinViewClin->MaladeRect.left+4 ;

if ( (_Spt = SC[5].First) != NULL)
 {
  for(_j=0;_j<_i;_j++)_Spt=_Spt->Next;


    do {
          c1= _Spt->Qualif;
          for(num=1;c1!=0;c1>>=1) {
           if(c1&1)
            {
             sprintf (_buf1,"%d",num);
             strcat (_buf1,".") ;
             }
             num++;
           }
          _width = GetWindowObjectDim ( win, OBJECT_GENERAL_SYMPTOM_LANG_1, _buf1,
                                  &_leading, &_ascent, &_descent );
           DisplayWindowObject ( win, &_from, &_to,OBJECT_GENERAL_SYMPTOM_LANG_1, _buf1,
                                  "", ATTRIBUTE_NORMAL );

           _from2.h = _from.h;
           _from.h = _from2.h +2+_width;

           _width = GetWindowObjectDim ( win, OBJECT_GENERAL_SYMPTOM_LANG_1,
                                          _Spt->SptTextLg1,
                                    &_leading, &_ascent, &_descent );
            DisplayWindowObject ( win, &_from, &_to,OBJECT_GENERAL_SYMPTOM_LANG_1,
                                  _Spt->SptTextLg1,
                                  "", ATTRIBUTE_NORMAL );
           _st++;
           _from2.v = _from.v;
           _from.v = _from2.v +14;
           _from.h = WinViewClin->MaladeRect.left+4 ;
```

```
                    _Spt = _Spt->Next;
                  }
                while((_Spt!=NULL)&&(_st<4));

                if (mode == CLIN_REFRESH_1)
                WinViewClin->LastSpt_1 = _st;

                if ( mode != CLIN_REFRESH_1 ) {
                    if ( WinViewClin->FirstSpt_1 == 0 ) {
                            SetWindowScrollBar ( xvt_win_get_ctl ( win, WIN_CCASE_VSCROLL_1 ),
                                                 HVSCROLL, 1L, (long)SC[5].NbSpt,
                                                        (long) 1 );
                    }
                    else {
                            SetWindowScrollBar ( xvt_win_get_ctl ( win, WIN_CCASE_VSCROLL_1 ),
                                                 HVSCROLL, 1L, (long)SC[5].NbSpt,
                                                 (long)( WinViewClin->FirstSpt_1 ) );
                    }
                }
            }
  }

/*********************************************/
Cette fonction parcourt les clipboards localisation
Sensation, modalité et enregistre les symptoms dans
un clipboard tampon
/*********************************************/

 void   CalculateSymptomMaladie( )
      {
         SYMPTOM_IN_MEMORY *_Spt;
         int  _scnumber,
              _num;
         unsigned char c1 =0;

         FreeSptInMemory(4,0);
         for (_num =1; _num<9;_num++)
            {
             for (_scnumber=0; _scnumber<3; _scnumber++)
                {
                 if ( (_Spt = SC[_scnumber].First) != NULL)
                    {
                      do
                        {
                          c1 = 1;
                          c1 = c1 << (_num-1);
                          if (_Spt->Qualif & c1)
                          CopySptToClinique((int)4,(unsigned char)c1,_Spt);  /* 1 pcq il faut qlc*/
                         _Spt = _Spt->Next;
```

```
                    }
                   while ( (_Spt != NULL));
                  }
               }
            }
         }


/*********************************************/
Cette fonction parcourt le clipboard de concomitants
Et enregistre les symptômes dans un clipboard tampon
/*********************************************/


 void   CalculateSymptomMalade( )
      {
         SYMPTOM_IN_MEMORY *_Spt;
         int  _num;
         unsigned char c1 =0;

         FreeSptInMemory(5,0);
         for (_num =1; _num<9;_num++)
            {
               if ( (_Spt = SC[3].First) != NULL)
                  {
                    do
                      {
                        c1 = 1;
                        c1 = c1 << (_num-1);
                        if (_Spt->Degrees & c1)
                        CopySptToClinique((int)5,(unsigned char)c1,_Spt);  /* 1 pcq il faut qlc*/
                       _Spt = _Spt->Next;
                       }
                    while ( (_Spt != NULL));
                  }
               }
         }

/*********************************************/

/*********************************************/

 void   CalculateSymptomCase( )
      {
         SYMPTOM_IN_MEMORY *_Spt;
         int  _scnumber,
              _num;
         unsigned char c1 =0;

         FreeSptInMemory(4,0);
```

```
        for (_num =1; _num<9;_num++)
           {
            for (_scnumber=0; _scnumber<4; _scnumber++)
               {
                if ( (_Spt = SC[_scnumber].First) != NULL)
                   {
                     do
                       {
                         c1 = 1;
                         c1 = c1 << (_num-1);
                         if (_Spt->Qualif & c1)
                           {
                            CopySptToQualif((int)4,_scnumber,c1,_Spt);
                            }
                         if (_Spt->Degrees & c1)
                           {
                            CopySptToDegrees((int)4,_scnumber,c1,_Spt);
                            }
                        _Spt = _Spt->Next;
                        }
                     while ( (_Spt != NULL));
                    }
               }
           }
      }


/*************************************************************/
Copie dans un clipboard tampon les symptômes du cas
/*************************************************************/
int
CopySptToClinique (_SCNumber,_type, _spt )
int _SCNumber;
unsigned char _type;
SYMPTOM_IN_MEMORY    *_spt;
{
  int                 _i,
                      _j;

           /*************************************************************/
           /* Allocating a structure to take this symptom in memory */
           /*************************************************************/
           if ( !(SptInMemory =
              (SYMPTOM_IN_MEMORY *) Malloc (sizeof (SYMPTOM_IN_MEMORY)) ) ) {
              xvt_dm_post_note( GetMessage ( FileMessage, 84, Option.General.Lang, &RecMessage) );
              return ( RADAR_NOT_OK );
           }
           SptInMemory->Next = (SYMPTOM_IN_MEMORY *)NULL;
#if DEBUG
           SptInMemory->Check = DB_CHECK;
```

```
#endif
        /*********************************************/
        /* Allocation of memory for the symptom texts */
        /*********************************************/

        if ( !(SptInMemory->SptTextLg1 =
           (unsigned char *) Malloc ( strlen (_spt->SptTextLg1) + 2 )) )   {
           xvt_dm_post_note( GetMessage ( FileMessage, 88, Option.General.Lang, &RecMessage) );
           Free ( (unsigned char *) SptInMemory);
           return (RADAR_NOT_OK);
        }

        if ( !(SptInMemory->SptTextLg2 =
           (unsigned char *) Malloc ( strlen (_spt->SptTextLg2) + 2)) )   {
           xvt_dm_post_note( GetMessage ( FileMessage, 89, Option.General.Lang, &RecMessage) );
           Free ( (unsigned char *) SptInMemory->SptTextLg1);
           Free ( (unsigned char *) SptInMemory);
           return (RADAR_NOT_OK);
        }

        memset (SptInMemory->SptTextLg1, '\0', strlen (_spt->SptTextLg1));
        memset (SptInMemory->SptTextLg2, '\0', strlen (_spt->SptTextLg2));

        /*********************************************/
        /* On alloue de la memoire pour les remedes   */
        /*********************************************/

        _i =  (int) _spt->NbRemedies;
        _i += 2;

        if ( !(SptInMemory->RmdList =
             (REMED_LIST *) Malloc ( (sizeof (REMED_LIST) * (_i)) ))  )   {
           xvt_dm_post_note( GetMessage ( FileMessage, 90, Option.General.Lang, &RecMessage) );
           Free ( (unsigned char *) SptInMemory->Spt TextLg1);
           Free ( (unsigned char *) SptInMemory->SptTextLg2);
           Free ( (unsigned char *) SptInMemory);
           return (RADAR_NOT_OK);
        }

        /*********************************************/
        /* On recopie le symptome membres ... membres */
        /*********************************************/
        SptInMemory->Intensity  =  _spt->Intensity;
        SptInMemory->Group      =  _spt->Group     ;
        SptInMemory->Degrees    =  (unsigned char)_type   ;
        SptInMemory->Qualif     =  (unsigned c har)_type   ;
        SptInMemory->View       =  _spt->View      ;
        SptInMemory->NbRemedies =  _spt->NbRemedies;
        SptInMemory->Origin     =  _spt->Origin    ;
        SptInMemory->Langue[0]  =  _spt->Langue[0] ;
```

```
            SptInMemory->Langue[1]  =  _spt->Langue[1] ;

            strcpy( SptInMemory->SptTextLg1, _spt->SptTextLg1 );
            strcpy( SptInMemory->SptTextLg2, _spt->SptTextLg2 );

            for (_j=0; _j<(int)_spt->NbRemedies; _j++) {
               SptInMemory->RmdList[_j].Check    = DB_CHECK;
               SptInMemory->RmdList[_j].RemedId  = (unsigned short) _spt->RmdList [_j].RemedId;
               SptInMemory->RmdList[_j].Degree   = (unsigned char)  _spt->RmdList [_j].Degree;
               SptInMemory->RmdList[_j].AuthorId = (unsigned short) _spt->RmdList [_j].AuthorId;
            }
            SptInMemory->Next = (SYMPTOM_IN_MEMORY *)NULL;

            UpdateSymptomClipboard ( _SCNumber+1, SptInMemory->NbRemedies );

   return ( RADAR_OK );
}

/*****************************************************************************/
Copie le symptôme avec son numéro de symptôme complet
/*****************************************************************************/
int
CopySptToQualif (_SCNumber,_type, c1, _spt )
int _SCNumber,_type;
unsigned char c1;
SYMPTOM_IN_MEMORY    *_spt;
{
  int                _i,
                     _j;

            /*********************************************************/
            /* Allocating a structure to take this symptom in memory */
            /*********************************************************/
            if ( !(SptInMemory =
               (SYMPTOM_IN_MEMORY *) Malloc (sizeof (SYMPTOM_IN_MEMORY)) ) ) {
               xvt_dm_post_note( GetMessage ( FileMessage, 84, Option.General.Lang, &RecMessage) );
               return ( RADAR_NOT_OK );
            }
            SptInMemory->Next = (SYMPTOM_IN_MEMORY *)NULL;
#if DEBUG
            SptInMemory->Check = DB_CHECK;
#endif
            /*********************************************/
            /* Allocation of memory for the symptom texts */
            /*********************************************/

            if ( !(SptInMemory->SptTextLg1 =
               (unsigned char *) Malloc ( strlen (_spt->SptTextLg1) + 2 )) )    {
               xvt_dm_post_note( GetMessage ( FileMessage, 88, Option.General.Lang, &RecMessage) );
               Free ( (unsigned char *) SptInMemory);
```

```
    return (RADAR_NOT_OK);
}

if ( !(SptInMemory->SptTextLg2 =
    (unsigned char *) Malloc ( strlen (_spt->SptTextLg2) + 2)) )   {
    xvt_dm_post_note( GetMessage ( FileMessage, 89, Option.General.Lang, &RecMessage) );
    Free ( (unsigned char *) SptInMemory->SptTextLg1);
    Free ( (unsigned char *) SptInMemory);
    return (RADAR_NOT_OK);
}

memset (SptInMemory->SptTextLg1, '\0', strlen (_spt->SptTextLg1));
memset (SptInMemory->SptTextLg2, '\0', strlen (_spt->SptTextLg2));

/*********************************************/
/* On alloue de la memoire pour les remedes   */
/*********************************************/

_i =  (int) _spt->NbRemedies;
_i += 2;

if ( !(SptInMemory->RmdList =
     (REMED_LIST *) Malloc ( (sizeof (REMED_LIST) * (_i)) ))  )   {
    xvt_dm_post_note( GetMessage ( FileMessage, 90, Option.General.Lang, &RecMessage) );
    Free ( (unsigned char *) SptInMemory->SptTextLg1);
    Free ( (unsigned char *) SptInMemory->SptTextLg2);
    Free ( (unsigned char *) SptInMemory);
    return (RADAR_NOT_OK);
}

/*******************************************/
/* On recopie le symptome membres ... membres */
/*******************************************/
SptInMemory->Intensity  =  (char)_type     ;
/*SptInMemory->Intensity  =  _spt->SCNumber   ;*/
SptInMemory->Group      =  _spt->Group     ;
SptInMemory->Degrees    =  _spt->Degrees   ;
SptInMemory->Qualif     =   c1             ;
SptInMemory->View       =  _spt->View      ;
SptInMemory->NbRemedies =  _spt->NbRemedies;
SptInMemory->Origin     =  _spt->Origin    ;
SptInMemory->Langue[0]  =  _spt->Langue[0] ;
SptInMemory->Langue[1]  =  _spt->Langue[1] ;

strcpy( SptInMemory->SptTextLg1, _spt->SptTextLg1 );
strcpy( SptInMemory->SptTextLg2, _spt->SptTextLg2 );

for (_j=0; _j<(int)_spt->NbRemedies; _j++) {
    SptInMemory->RmdList[_j].Check    = DB_CHECK;
    SptInMemory->RmdList[_j].RemedId  = (unsigned short) _spt->RmdList [_j].RemedId;
```

```
            SptInMemory->RmdList[_j].Degree   = (unsigned char)  _spt->RmdList [_j].Degree;
            SptInMemory->RmdList[_j].AuthorId = (unsigned short) _spt->RmdList [_j].AuthorId;
         }
         SptInMemory->Next = (SYMPTOM_IN_MEMORY *)NULL;

         UpdateSymptomClipboard ( _SCNumber+1, SptInMemory->NbRemedies );

   return ( RADAR_OK );
}

/*****************************************************************************/
Copie le symptôme avec son numéro de symptôme complet (il s'agit des concomitants)
/*****************************************************************************/
int
CopySptToDegrees (_SCNumber,_type,c1, _spt )
int _SCNumber,_type;
unsigned char c1;
SYMPTOM_IN_MEMORY    *_spt;
{
  int                 _i,
                      _j;

         /*********************************************************/
         /* Allocating a structure to take this symptom in memory */
         /*********************************************************/
         if ( !(SptInMemory =
            (SYMPTOM_IN_MEMORY *) Malloc (sizeof (SYMPTOM_IN_MEMORY)) ) ) {
            xvt_dm_post_note( GetMessage ( FileMessage, 84, Option.General.Lang, &RecMessage) );
            return ( RADAR_NOT_OK );
         }
         SptInMemory->Next = (SYMPTOM_IN_MEMORY *)NULL;
#if DEBUG
         SptInMemory->Check = DB_CHECK;
#endif
         /*********************************************/
         /* Allocation of memory for the symptom texts */
         /*********************************************/

         if ( !(SptInMemory->SptTextLg1 =
            (unsigned char *) Malloc ( strlen (_spt->SptTextLg1) + 2 )) )   {
            xvt_dm_post_note( GetMessage ( FileMessage, 88, Option.General.Lang, &RecMessage) );
            Free ( (unsigned char *) SptInMemory);
            return (RADAR_NOT_OK);
         }

         if ( !(SptInMemory->SptTextLg2 =
            (unsigned char *) Malloc ( strlen (_spt->SptTextLg2) + 2)) )   {
            xvt_dm_post_note( GetMessage ( FileMessage, 89, Option.General.Lang, &RecMessage) );
            Free ( (unsigned char *) SptInMemory->SptTextLg1);
            Free ( (unsigned char *) SptInMemory);
```

```
            return (RADAR_NOT_OK);
        }

        memset (SptInMemory->SptTextLg1, '\0', strlen (_spt->SptTextLg1));
        memset (SptInMemory->SptTextLg2, '\0', strlen (_spt->SptTextLg2));

        /*********************************************/
        /* On alloue de la memoire pour les remedes  */
        /*********************************************/

        _i =  (int) _spt->NbRemedies;
        _i += 2;

        if ( !(SptInMemory->RmdList =
             (REMED_LIST *) Malloc ( (sizeof (REMED_LIST) * (_i)) ))  )   {
            xvt_dm_post_note( GetMessage ( FileMessage, 90, Option.General.Lang, &RecMessage) );
            Free ( (unsigned char *) SptInMemory->SptTextLg1);
            Free ( (unsigned char *) SptInMemory->SptTextLg2);
            Free ( (unsigned char *) SptInMemory);
            return (RADAR_NOT_OK);
        }

        /*******************************************/
        /* On recopie le symptome membres ... membres */
        /*******************************************/
        SptInMemory->Intensity  =  (char)_type     ;
        /*SptInMemory->Intensity  =  _spt->SCNumber   ;*/
        SptInMemory->Group      =  _spt->Group     ;
        SptInMemory->Degrees    =   c1             ;
        SptInMemory->Qualif     =  _spt->Qualif    ;
        SptInMemory->View       =  _spt->View      ;
        SptInMemory->NbRemedies =  _spt->NbRemedies;
        SptInMemory->Origin     =  _spt->Origin    ;
        SptInMemory->Langue[0]  =  _spt->Langue[0] ;
        SptInMemory->Langue[1]  =  _spt->Langue[1] ;

        strcpy( SptInMemory->SptTextLg1, _spt->SptTextLg1 );
        strcpy( SptInMemory->SptTextLg2, _spt->SptTextLg2 );

        for (_j=0; _j<(int)_spt->NbRemedies; _j++) {
            SptInMemory->RmdList[_j].Check    = DB_CHECK;
            SptInMemory->RmdList[_j].RemedId  = (unsigned short) _spt->Rm dList [_j].RemedId;
            SptInMemory->RmdList[_j].Degree   = (unsigned char)  _spt->RmdList [_j].Degree;
            SptInMemory->RmdList[_j].AuthorId = (unsigned short) _spt->RmdList [_j].AuthorId;
        }
        SptInMemory->Next = (SYMPTOM_IN_MEMORY *)NULL;

        UpdateSymptomClipboard ( _SCNumber+1, SptInMemory->NbRemedies );

    return ( RADAR_OK );
```

```
}
```

A.3.4.1 FONCTION X_CLIN

```
/*
        This file was generated by XVT-Design 3.03, a product of:

                XVT Software Inc.
                4900 Pearl East Circle
                Boulder, CO USA 80301
                303-443-4223, fax 303-443-0969

        Generated on Mon Mar 24 12:36:19 1997
*/

#include "xvt.h"
#include "xvtcm.h"
#include "x_radar.h"

/*
        Information about the window
*/
#define WIN_RES_ID WIN_CCASE
#define WIN_FLAGS 0x882L
#define WIN_CLASS ""
#define WIN_BORDER W_DOC
# include "radar.h"

/*
        Handler for window WIN_CCASE ("Cas Clinique")
*/
long XVT_CALLCONV1
#if XVT_CC_PROTO
WIN_CCASE_eh XVT_CALLCONV2 (WINDOW xdWindow, EVENT *xdEvent)
#else
WIN_CCASE_eh XVT_CALLCONV2 (xdWindow, xdEvent)
WINDOW xdWindow;
EVENT *xdEvent;
#endif
{
        short xdControlId = xdEvent->v.ctl.id;
                static WIN_MSG WinMsgClinique [] = {
                        WIN_CCASE,                          -1,    '\0',
                        WIN_CCASE_OK,                      875,    '\0',
                        WIN_CCASE_CANCEL,                  876,    '\0',
                        WIN_CCASE_MALE,                   4033,    '\0',
                        WIN_CCASE_FEMELLE,                4034,    '\0',
                        0,                      0,      '\0' };

        switch (xdEvent->type) {
```

```
case E_CREATE:
        /*
                Window has been created; first event sent to newly-created
                window.
        */
        {
                        LoadMessagesForWindow ( xdWindow, (WIN_MSG *) WinMsgClinique );
        }
        break;
case E_DESTROY:
        /*
                Window has been closed; last event sent to window.
        */
        xdRemoveHelpAssoc( xdWindow );
        {
        }
        break;
case E_FOCUS:
        {
        /*
                Window has lost or gained focus.
        */
        if (xdEvent->v.active)  {
                /*
                        Window has gained focus
                */
        } else {
                /*
                        Window has lost focus
                */
        }
        }
        break;
case E_SIZE:
        /*
                Size of window has been set or changed; sent when window  is
                created or subsequently resized by user or via xvt_vobj_move.
        */
        {
        }
        break;
case E_UPDATE:
        /*
                Window requires updating.
        */
        {
                        UpdateViewCliniqueWindow(xdWindow);
        }
        break;
case E_CLOSE:
```

```
        /*
                Request to close window; user operated "close" menu item on
                window system menu, or operated "close" control on window
                frame. Not sent if Close on File menu is issued. Window not
                closed unless xvt_vobj_destroy is called.
        */
        {
        xvt_vobj_destroy(xdWindow);
        }
        break;
case E_CHAR:
        /*
                Character typed.
        */
        {
        }
        break;
case E_MOUSE_UP:
        /*
                Mouse was released
        */
        {
        }
        break;
case E_MOUSE_DOWN:
        /*
                Mouse was pressed
        */
        {
        }
        break;
case E_MOUSE_DBL:
        /*
                Mouse was double clicked
        */
        {
        }
        break;
case E_MOUSE_MOVE:
        /*
                Mouse was moved
        */
        {
        }
        break;
case E_HSCROLL:
        {
        /*
                Horizontal scrollbar on frame was operated
        */
```

```
            switch (xdEvent->v.scroll.what) {
            case SC_LINE_UP:
                    break;
            case SC_LINE_DOWN:
                    break;
            case SC_PAGE_UP:
                    break;
            case SC_PAGE_DOWN:
                    break;
            case SC_THUMB:
                    break;
            case SC_THUMBTRACK:
                    break;
            default:
                    break;
            }
            }
            break;
    case E_VSCROLL:
            {
            /*
                    Vertical scrollbar on frame was operated
            */
            switch (xdEvent->v.scroll.what) {
            case SC_LINE_UP:
                    break;
            case SC_LINE_DOWN:
                    break;
            case SC_PAGE_UP:
                    break;
            case SC_PAGE_DOWN:
                    break;
            case SC_THUMB:
                    break;
            case SC_THUMBTRACK:
                    break;
            default:
                    break;
            }
            }
            break;
    case E_COMMAND:
            /*
                    User issued command on window menu bar (menu bar at top of
                    screen for Mac/CH).
            */
            {
            /*
                    No menubar was associated with this window
            */
```

```
        }
        break;
case E_CONTROL:
        /*
                User operated control in window.
        */
        {

        switch(xdControlId) {
        case WIN_CCASE_EDIT_PRENOM:                     {
                /*
                        Edit control was operated.
                */
                if (xdEvent->v.ctl.ci.v.edit.focus_change) {
                        if (xdEvent->v.ctl.ci.v.edit.active) {
                                /*
                                        focus has entered the control
                                */
                        } else {
                                /*
                                        focus has left the control
                                */
                        }
                } else {
                        /*
                                Contents of control were changed
                        */
                }
                }
                break;
        case WIN_CCASE_EDIT_NOM:                        {
                /*
                        Edit control was operated.
                */
                if (xdEvent->v.ctl.ci.v.edit.focus_change) {
                        if (xdEvent->v.ctl.ci.v.edit.active) {
                                /*
                                        focus has entered the control
                                */
                        } else {
                                /*
                                        focus has left the control
                                */
                        }
                } else {
                        /*
                                Contents of control were changed
                        */
                }
                }
```

```
        break;
case WIN_CCASE_EDIT_NAISSANCE:                  {
        /*
                Edit control was operated.
        */
        if (xdEvent->v.ctl.ci.v.edit.focus_change) {
                if (xdEvent->v.ctl.ci.v.edit.active) {
                        /*
                                focus has entered the control
                        */
                } else {
                        /*
                                focus has left the control
                        */
                }
        } else {
                /*
                        Contents of control were changed
                */
        }
        }
        break;
case WIN_CCASE_OK: /* "Sauver" */
        {
        }
        break;
case WIN_CCASE_CANCEL: /* "Annuler" */
        {
                xvt_vobj_destroy(xdWindow);
        }
        break;
case WIN_CCASE_VSCROLL_2:                       {
        /*
                Vertical scrollbar control was operated
        */

             long _pos;
        switch (xdEvent->v.ctl.ci.v.scroll.what) {
        case SC_LINE_UP:
                DisplayViewMaladieSymptom(xdWindow,CLIN_LINE_UP_2);
                break;
        case SC_LINE_DOWN:
                DisplayViewMaladieSymptom(xdWindow,CLIN_LINE_DOWN_2);
                break;
        case SC_PAGE_UP:
                break;
        case SC_PAGE_DOWN:
                break;
        case SC_THUMB:
                        _pos = GetWindowScrollBarThumbPos (
```

```
                                                    xvt_win_get_ctl ( xdWindow,
WIN_CCASE_VSCROLL_2),
                                                    HVSCROLL, 1L,
                                                    SC[4].NbSpt,
                                                    (int)xdEvent->v.ctl.ci.v.scroll.pos
);
                                                if ( _pos == 1l )    _pos = 0l;
                                                if ( _pos <   0l )    _pos = 0l;
                                                    if ( _pos >   (long) SC[4].NbSpt ) {
                                                        _pos = (long) SC[4].NbSpt;
                                                    }
                                                SetWindowScrollBar ( xvt_win_get_ctl ( xdWindow,
WIN_CCASE_VSCROLL_2 ),
                                                    HVSCROLL, 1L,
                                                    SC[4].NbSpt,
                                                    _pos );

                                                WinViewClin->FirstSpt_2      = (int) _pos;

        DisplayViewMaladieSymptom(xdWindow,CLIN_DUMMY_REFRESH_2);

                        break;
                case SC_THUMBTRACK:
                        _pos = GetWindowScrollBarThumbPos (
                                                    xvt_win_get_ctl ( xdWindow,
WIN_CCASE_VSCROLL_2),
                                                    HVSCROLL, 1L,
                                                    SC[4].NbSpt,
                                                    (int)xdEvent->v.ctl.ci.v.scroll.pos
);
                                                if ( _pos == 1l )    _pos = 0l;
                                                if ( _pos <   0l )    _pos = 0l;
                                                    if ( _pos >   (long) SC[4].NbSpt ) {
                                                        _pos = (long) SC[4].NbSpt;
                                                    }
                                                SetWindowScrollBar ( xvt_win_get_ctl ( xdWindow,
WIN_CCASE_VSCROLL_2 ),
                                                    HVSCROLL, 1L,
                                                    SC[4].NbSpt,
                                                    _pos );

                                                WinViewClin->FirstSpt_2      = (int) _pos;

        DisplayViewMaladieSymptom(xdWindow,CLIN_DUMMY_REFRESH_2);

                        break;
                default:
                        break;
                }
                }
                break;
            case WIN_CCASE_MALE:
                {
```

```
                        xdCheckRadioButton(xdWindow, WIN_CCASE_MALE,
                                WIN_CCASE_MALE, WIN_CCASE_FEMELLE);
                        }
                        break;
                case WIN_CCASE_FEMELLE:
                        {
                        xdCheckRadioButton(xdWindow, WIN_CCASE_FEMELLE,
                                WIN_CCASE_MALE, WIN_CCASE_FEMELLE);
                        }
                        break;
                case WIN_CCASE_VSCROLL_1:                       {
                        long  _pos;
                        /*
                                Vertical scrollbar control was operated
                        */
                        switch (xdEvent->v.ctl.ci.v.scroll.what) {
                        case SC_LINE_UP:
                                                        DisplayViewMaladeSymptom(xdWindow,CLIN_LINE_UP_1);
                                break;
                        case SC_LINE_DOWN:

        DisplayViewMaladeSymptom(xdWindow,CLIN_LINE_DOWN_1);
                                break;
                        case SC_PAGE_UP:
                                break;
                        case SC_PAGE_DOWN:
                                break;
                        case SC_THUMB:
                                                        _pos = GetWindowScrollBarThumbPos (
                                                                xvt_win_get_ctl ( xdWindow,
WIN_CCASE_VSCROLL_1),
                                                                HVSCROLL, 1L,
                                                                SC[5].NbSpt,
                                                                (int)xdEvent->v.ctl.ci.v.scroll.pos
);
                                                        if ( _pos == 1l )     _pos = 01;
                                                        if ( _pos <   01 )     _pos = 01;
                                                                if ( _pos >    (long) SC[5].NbSpt ) {
                                                                        _pos = (long) SC[5].NbSpt;
                                                                }
                                                        SetWindowScrollBar ( xvt_win_get_ctl ( xdWindow,
WIN_CCASE_VSCROLL_1 ),
                                                                HVSCROLL, 1L,
                                                                SC[5].NbSpt,
                                                                _pos );

                                                        WinViewClin->FirstSpt_1       = (int)  _pos;

        DisplayViewMaladeSymptom(xdWindow,CLIN_DUMMY_REFRESH_1);


                                break;
```

```
                        case SC_THUMBTRACK:
                                                        _pos = GetWindowScrollBarThumbPos (
                                                                xvt_win_get_ctl ( xdWindow,
WIN_CCASE_VSCROLL_1),
                                                                HVSCROLL, 1L,
                                                                SC[5].NbSpt,
                                                                (int)xdEvent->v.ctl.ci.v.scroll.pos
);
                                                if ( _pos == 1l )     _pos = 0l;
                                                if ( _pos <   0l )     _pos = 0l;
                                                        if ( _pos >    (long) SC[5].NbSpt ) {
                                                                        _pos = (long) SC[5].NbSpt;
                                                        }
                                                SetWindowScrollBar ( xvt_win_get_ctl ( xdWindow,
WIN_CCASE_VSCROLL_1 ),
                                                                HVSCROLL, 1L,
                                                                SC[5].NbSpt,
                                                                _pos );

                                                WinViewClin->FirstSpt_1       = (int) _pos;
        DisplayViewMaladeSymptom(xdWindow,CLIN_DUMMY_REFRESH_1);


                                break;
                        default:
                                break;
                        }
                        }
                        break;
                case WIN_CCASE_NOMDOSSIER:                      {
                        /*
                                Edit control was operated.
                        */
                        if (xdEvent->v.ctl.ci.v.edit.focus_change) {
                                if (xdEvent->v.ctl.ci.v.edit.active) {
                                        /*
                                                focus has entered the control
                                        */
                                } else {
                                        /*
                                                focus has left the control
                                        */
                                }
                        } else {
                                /*
                                        Contents of control were changed
                                */
                        }
                        }
                        break;
                case WIN_CCASE_EDIT_DATECONS:                   {
```

```
        /*
                Edit control was operated.
        */
        if (xdEvent->v.ctl.ci.v.edit.focus_change) {
                if (xdEvent->v.ctl.ci.v.edit.active) {
                        /*
                                focus has entered the control
                        */
                } else {
                        /*
                                focus has left the control
                        */
                }
        } else {
                /*
                        Contents of control were changed
                */
        }
        }
        break;
case WIN_CCASE_EDIT_DIAGNOS:                    {
        /*
                Edit control was operated.
        */
        if (xdEvent->v.ctl.ci.v.edit.focus_change) {
                if (xdEvent->v.ctl.ci.v.edit.active) {
                        /*
                                focus has entered the control
                        */
                } else {
                        /*
                                focus has left the control
                        */
                }
        } else {
                /*
                        Contents of control were changed
                */
        }
        }
        break;
case WIN_CCASE_EDIT_RESULT:                     {
        /*
                Edit control was operated.
        */
        if (xdEvent->v.ctl.ci.v.edit.focus_change) {
                if (xdEvent->v.ctl.ci.v.edit.active) {
                        /*
                                focus has entered the control
                        */
```

```
                        } else {
                                /*
                                        focus has left the control
                                */
                        }
                } else {
                        /*
                                Contents of control were changed
                        */
                }
                }
                break;
        case WIN_CCASE_VSCROLL_28: /* "Vertical Scrollbar 28" */
                {

                /*
                        Vertical scrollbar control was operated
                */
                TXEDIT     _txt = (TXEDIT)xvt_win_get_tx ( xdWindow, WIN_CCASE_REMED E );
                CLIN_WINDOW    *_clin = (CLIN_WINDOW *)xvt_vobj_get_data( xdWindow );
                T_LNUM   _org_line;
                switch (xdEvent->v.ctl.ci.v.scroll.what) {
                case SC_LINE_UP:
                        xvt_tx_scroll_vert ( _txt, 1 );
                        break;
                case SC_LINE_DOWN:
                        xvt_tx_scroll_vert ( _txt, -1 );
                        break;
                case SC_PAGE_UP:
                        xvt_tx_scroll_vert ( _txt, _clin->Line1 );
                        break;
                case SC_PAGE_DOWN:
                        xvt_tx_scroll_vert ( _txt, _clin->Line1 * -1 );
                        break;
                case SC_THUMB:
                        xvt_tx_get_origin ( _txt, NULL, NULL, &_org_ line, NULL );
                        xvt_tx_scroll_vert ( _txt, (int)_org_line - xdEvent-
>v.ctl.ci.v.scroll.pos );
                        break;
                case SC_THUMBTRACK:
                        break;
                default:
                        break;
                }
                }
                break;
        case WIN_CCASE_VSCROLL_29: /* "Vertical Scrollbar 29" */
                {

                /*
                        Vertical scrollbar control was operated
```

```
                        */
                        TXEDIT      _txt = (TXEDIT)xvt_win_get_tx ( xdWindow, WIN_CCASE_REMARQUE );
                        CLIN_WINDOW    *_clin = (CLIN_WINDOW *)xvt_vobj_get_data( xdWindow );
                        T_LNUM    _org_line;
                        switch (xdEvent->v.ctl.ci.v.scroll.what) {
                        case SC_LINE_UP:
                                xvt_tx_scroll_vert ( _txt, 1 );
                                break;
                        case SC_LINE_DOWN:
                                xvt_tx_scroll_vert ( _txt, -1 );
                                break;
                        case SC_PAGE_UP:
                                xvt_tx_scroll_vert ( _txt, _clin->Line2 );
                                break;
                        case SC_PAGE_DOWN:
                                xvt_tx_scroll_vert ( _txt, _clin->Line2 * -1 );
                                break;
                        case SC_THUMB:
                                xvt_tx_get_origin ( _txt, NULL, NULL, &_org_line, NULL );
                                xvt_tx_scroll_vert ( _txt, (int)_org_line - xdEvent-
>v.ctl.ci.v.scroll.pos );
                                break;
                        case SC_THUMBTRACK:
                                break;
                        default:
                                break;

                        }
                        }
                        break;
                default:
                        break;
                }
                }
                break;
        case E_FONT:
                /*
                        User issued font command on window menu bar (menu bar at top of
                        screen for Mac/CH).
                */
                {
                }
                break;
        case E_TIMER:
                /*
                        Timer associated with window went off.
                */
                {
                }
                break;
        case E_USER:
```

```
                /*
                        Application initiated.
                */
                {
                switch (xdEvent->v.user.id) {
                case -1:
                default:
                        break;
                }
                }
                break;
        default:
                break;
        }
        xvt_tx_process_event(xdWindow, xdEvent);
        return 0L;
}
```